МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

НЕВИННОМЫССКИЙ ТЕХНОЛОГИЧЕСКИЙ ИНСТИТУТ (ФИЛИАЛ)

Методические указания к выполнению лабораторных работ

для студентов направления

15.04.04 «Автоматизация технологических процессов и производств»

по дисциплине

«ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ УПРАВЛЕНИЯ»

Невинномысск, 2023

Методические указания предназначены для студентов направления 15.04.04 – «Автоматизация технологических процессов» и других технических специальностей и направлений. Они содержат основы теории, описание опытных установок, порядок проведения лабораторных работ и обработки экспериментальных данных, перечень контрольных вопросов для самоподготовки и список рекомендуемой литературы. Работы подобраны и расположены в соответствии с методикой изучения дисциплины «Интеллектуальные системы управления». Объем и последовательность выполнения работ определяются преподавателем в зависимости от количества часов, предусмотренных учебным планом дисциплин.

Методические указания разработаны в соответствии с требованиями ФГОС ВО в части содержания и уровня подготовки выпускников направления 15.04.04 – «Автоматизация технологических процессов».

Содержание

Введение 4

Лабораторная работа №1. Часть 1. Семантические сети 5

Лабораторная работа №1 Часть 2. Использование фреймов для представления знаний 7

Лабораторная работа № 1. Часть 3. Описание предметной области. Разработка базы фактов и правил интеллектуальной системы 9

Лабораторная работа № 2. Часть 1. Использование правил продукции для представления знаний. прямая цепочка рассуждений 12

Лабораторная работа № 2. Часть 2. Использование правил продукции для представления знаний обратная цепочка рассуждений 14

Лабораторная работа № 3. Использование Simulink при построении нейронных сетей. 17

Лабораторная работа № 4. Графический интерфейс гибридных систем 31

Лабораторная работа № 5. Часть 1. Использование графического интерфейса для построения нечеткой аппроксимирующей системы 34

Лабораторная работа № 5. Часть 2. Создание пользовательских функций принадлежности 37

Лабораторная работа № 6. Часть 1. Нечеткие множества. операции над нечеткими множествами 39

Лабораторная работа №6. Часть 2. проектирование системы нечеткого вывода 43

Лабораторная работа № 7. Построение нечеткой аппроксимирующей системы в пакете Fuzzy Logic Toolbox 49

Лабораторная работа №8. Моделирование нечеткой системы управления вентилятором 56

Лабораторная работа № 8 Нечеткая экспертная система с алгоритмом вывода Mamdani 62

Лабораторная работа № 9. Часть 1. Работа Fuzzy Logic с блоками Simulink 63

Лабораторная работа № 9. Часть 2. Разработка гибридных интеллектуальных систем в среде Matlab 66

Введение

Целью курса "Интеллектуальные системы управления" является ознакомление студентов с проблематикой представления знаний в информационных системах, областями использования систем искусственного интеллекта, применение интеллектуальных систем в системах управления, освещение теоретических и организационно-методических вопросов построения и функционирования систем обработки знаний, привитие навыков практических работ по проектированию интеллектуальных систем управления.

Задачи изучения дисциплины:

- приобретение студентами знаний и практических навыков в области, определяемой основной целью дисциплины;
- освоить методы модернизации и автоматизации действующих и проектирование новых автоматизированных и автоматических производственных и технологических процессов с использованием автоматизированных систем на базе интеллектуальных технологий;
- совершенствование способности к абстрактному мышлению, анализу и синтезу;
- научиться проводить математическое моделирование процессов, оборудования, средств и систем автоматизации, контроля, диагностики и управления с использованием современных интеллектуальных систем и технологий.

Код, формулировка компетенции	Код, формулировка индикатора	Планируемые результаты обучения по дисциплине (модулю), характеризующие этапы формирования компетенций, индикаторов
ОПК-5. Способен разрабатывать аналитические и численные методы при создании математических моделей машин, приводов, оборудования, систем, технологических процессов	ИД-1$_{ОПК-5}$ Использует аналитические и численные методы для получения математических моделей машин, приводов, оборудования, систем, технологических процессов	Использует современное программное и аппаратное обеспечение автоматизированных систем, аналитические и численные методы идентификации машин, приводов, оборудования, систем, технологических процессов.
	ИД-2$_{ОПК-5}$ Создает математические модели машин, приводов, оборудования, систем, технологических процессов	Применяет аналитико-численные методы и комплексы программ для получения математических моделей и исследования машин, приводов, оборудования, систем, технологических процессов

Лабораторная работа №1. Часть 1. Семантические сети

Цель работы: Научиться использовать семантические сети для представления знаний в интеллектуальных системах.

1. Теоретическая часть

Семантическая сеть – это один из способов представления знаний. Изначально семантическая сеть была задумана как модель представления долговременной памяти в психологии, но впоследствии стала одним из способов представления знаний в ЭС.

Семантика – означает общие отношения между символами и объектами из этих символов.

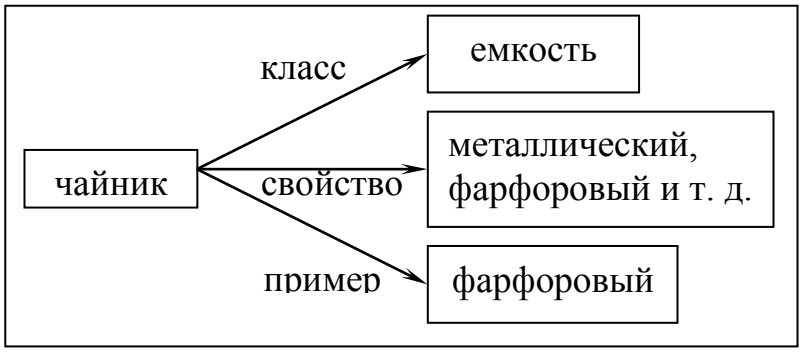


Рис.1. Простейший образец семантической сети.

Вершины – это объекты, дуги – это отношения. Семантическая модель не раскрывает сама по себе каким образом осуществляется представление знаний. Поэтому семантическая сеть рассматривается как метод представления знаний и структурирования знаний. При расширении семантической сети в ней возникают другие отношения:

IS –A (принадлежит) и PART OF (является частью) отношение:

целое → часть.

Ласточка IS – A птица, «нос» PART OF «тело». Например:

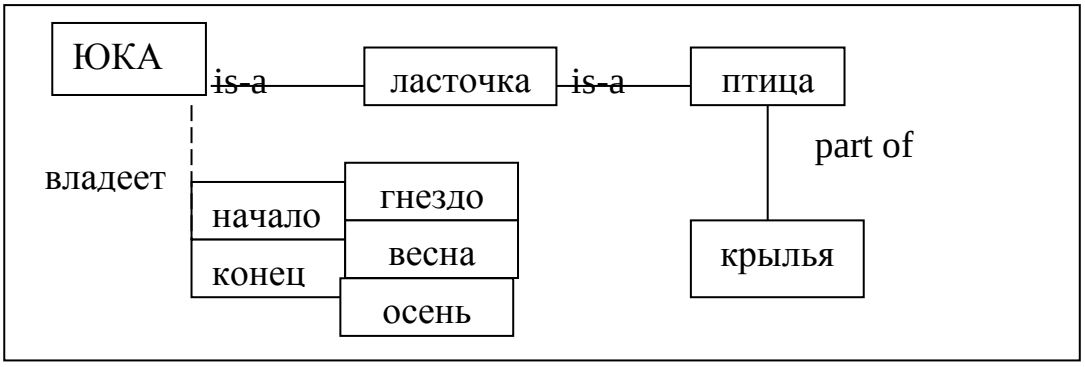


Рис.2. Расширение семантической сети

Могут быть и другие отношения: владеет. Тогда семантическая сеть расширяется иерархически (вершина имеет две ветви). Кроме того, можно расширить сеть и другим отношением:

период → «весна – лето».

Получается иерархическая структура понятия ЮКО. Можно разбить на подсхемы. Большой проблемой для семантических сетей является то, что результат вывода не гарантирует достоверности, так как вывод есть просто наследование свойств ветви is-a.

Для отображения иерархических отношений между объектами и введения единой семантики в семантические сети было предложено использовать процедурные сети. Сеть строится на основе класса (понятия); вершины, дуги и процедуры представлены как объекты.

Варианты лабораторной работы №1

Вариант 1

Компания, в которой Вы работаете, получила задание на разработку справочной системы по журналам издательства «Издательство Мечты». Данная компания выпускает различные по целевой аудитории, ценовой категории и объему страниц журналы.

Вам необходимо построить модуль на основе семантической сети, позволяющий определить целевую аудиторию для различных журналов, а также для кого предназначено издание и его стоимость. Ваша задача построить семантическую сеть на основе информации, представленной в таблице 1.

Таблица 1

Название журнала	**Основная целевая аудитория**	**Стоимость одного номера, руб.**	**Объем страниц журнала**	**Какая информация представлена в журнале**	**Возможна ли подписка на журнал**
Тюниг автомобилей	Мужчины	140	170	Современные технологии тюнинга автомобилей	нет
Мода	Женщины	90	90	Новейшие тенденции моды	да
Компьютерные и видео игры	Мужчины и женщины	65	60	Все о компьютерных и видео играх	нет
Рукоделие	Женщины	45	50	Эксклюзивные вещи своими руками	да
Фотография	Мужчины и женщины	100	95	Основы и секреты фотографии	да
Кино и музыка	Мужчины и женщины	30	30	Только актуальная информация и кино и музыке	нет

В построенной семантической сети определить:

1. Какой журнал предоставляет информацию о современных технологии тюнинга автомобилей?
2. Какие журналы предназначены для мужчин?
3. Какие журналы стоят 100 рублей?
4. На какие журналы можно оформить подписку?

Вариант 2

Ваша задача состоит в создании экспертной системы АСУ предприятия, автоматизирующей контроль за выполнением задач коллективом предприятия. АСУ следует построить в виде семантической сети.

Система должна описывать структуру предприятия, в том числе руководство и структуру отделов.

Система так же должна описывать выполняемые предприятием задания, в том числе:

1. Наименование задания.
2. Сроки его выполнения.
3. Этапы выполнения задания и их очередность.

Для каждого этапа описывается:

1. Отдел, выполняющий этап.
2. Ответственное лицо, обычно – руководитель отдела или подразделения.
3. Сроки начала и окончания этапа.

Предприятие, для которого строится система – ООО «Созвездие»:

Директор: Иванов И.И.

Отдел разработки, нач. отдела – Перов П.П.

В составе отдела разработки:

Бюро постановки задач, нач. бюро – Сидоров С.С.
Бюро программирования, нач. бюро – Брайан Керниган
Бюро сопровождения, нач. бюро – Билл Гейтс
Отдел маркетинга, нач. отдела – Тошико Ямада

Задания в работе:

1. Разработка текстового редактора «Созвездие», этапы – постановка задачи, программирование, продвижение на рынок, поддержка.
2. Разработка Интернет - браузера «Созвездие», этапы – постановка задачи, программирование, продвижение на рынок, поддержка.

В построенной семантической сети определить:

1. Кто является начальником отдела маркетинга?
2. Какие задания выполняет ООО «Созвездие»?
3. Чем занимается Иванов И.И.?
4. Какие сроки выполнения заданы для разработки Интернет - браузера?

Контрольные вопросы

1. Что такое семантическая сеть и для чего ее применяют?
2. В чем состоит идея создания семантической сети?
3. Каким образом представляются данные в семантической сети?
4. Существуют ли ограничения на число связей элементов, свойств и сложность при построении семантической сети?
5. Какие отношения предложены в качестве операторов отношения для группировки вершин?

Лабораторная работа №1 Часть 2. Использование фреймов для представления знаний

Цель работы: Научиться использовать фреймы для представления знаний в интеллектуальных системах

1. Теоретическая часть

Фреймы - один из распространенных формализмов представления знаний в ЭС. Фрейм можно представить себе как структуру, состоящую из набора ячеек - слотов. Каждый слот состоит из имени и ассоциируемых с ним значений. Значения могут представлять собой данные, процедуры, ссылки на другие фреймы или быть пустыми. Такое построение оказывается очень удобным для моделирования аналогий, описания областей с родовидовыми связями понятий и т.п.

Любой фрейм состоит из некоторых составляющих, имена и содержание которых описано ниже:

1. Имя фрейма. Это идентификатор, присваиваемый фрейму, фрейм должен иметь имя уникальное в данной фреймовой системе.
2. Имя слота. Это идентификатор, присваиваемый слоту; слот должен иметь уникальное имя во фрейме, к которому он принадлежит. Обычно имя слота не несет никакой смысловой нагрузки и является лишь идентификатором данного слота.
3. Указатели наследования. Эти указатели касаются только фреймовых систем иерархического типа, основанные на отношениях “абстрактное-конкретное”, они показывают, какую информацию об атрибутах слотов во фрейме верхнего уровня наследуют слоты с такими же именами во фрейме нижнего уровня. Типичные указатели наследования Unique (U: - уникальный), Same (S: такой же), Range (R: установление границ), Override (O: игнорировать) и т.п. U показывает, что фрейм может иметь слоты с разными значениями: S - все слоты должны иметь одинаковые значе-

ния, R - значение слотов фрейма нижнего уровня должны находиться в пределах, указанных значениями слотов фрейма верхнего уровня, O - при отсутствии указания значение слота фрейма верхнего уровня становится значением слота фрейма нижнего уровня, но в случае определения нового значения слотов фреймов нижних уровней указываются в качестве значений слотов.

4. Указание типа данных. указывается, что слот имеет численное значение, либо служит указателем другого фрейма. К типам данных относятся:

FRAME (указатель), INTEGER (целый), REAL (действительный), BOOL (булев), LISP (присоединенная процедура), TEXT (текст), LIST (список), TABLE (таблица), EXPRESSION (выражение) и др.

5. Значение слота. Пункт ввода значения слота. Значение слота должно совпадать с указанным типом данных этого слота, кроме того должно выполняться условие наследования.

6. Демон. Здесь дается определение демонов типа IF-NEEDED, IF-ADDED, IF-REMOVED и т.д. Демоном называется процедура, автоматически запускаемая при выполнении некоторого условия. демоны запускаются при обращении к соответствующему слоту. Кроме того, демон является разновидностью присоединенной процедуры.

7. Присоединенная процедура. В качестве значения слота можно использовать программу процедурного типа. Когда мы говорим, что в моделях представления знаний фреймами объединяются процедурные и декларативные знания, то считаем демоны и присоединенные процедуры процедурными знаниями.

Особенностью иерархической структуры является то, что информация об атрибутах фрейма на верхнем уровне совместно используется всеми фреймами нижних уровней, связанных с ним.

Например: Фреймовое представление конференции.

Иерархические фреймовые структуры базируются на отношениях IS – A между фреймами, описывающими некоторую конференцию. Все фреймы должны содержать информацию о ДАТЕ, МЕСТЕ, НАЗВАНИИ ТЕМЫ, ДОКЛАДЧИКЕ. Таким образом, на самом верхнем уровне определен фрейм КОНФЕРЕНЦИЯ.

Конференции разделяются на коммерческие и по развитию. Они составляют дочерние фреймы. В них могут быть добавлены слоты: объем торговли и бюджет**.**

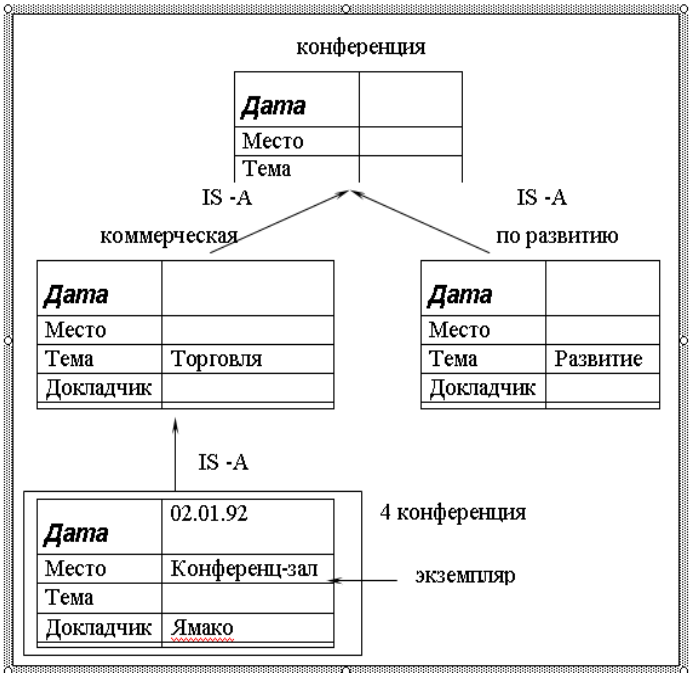


Рис.3. Пример фреймовой модели

2. Порядок выполнения работы:

1. Изучить теоретическую часть по приведенным выше данным и дополнительной литературе.

2. Просмотреть демонстрационный пример.

3. Получить у преподавателя вариант задания для выполнения.
4. Построить фреймовую модель заданного объекта;
5. Реализовать программу с использованием фреймовой модели

3. Варианты заданий

Используя фреймовую модель представления знаний реализовать структуру отношений, описывающие следующие ситуации:

1. экзамен по дисциплине за семестр у преподавателя при составляющих: семестр, экзамен, преподаватель, оценка, студент, получать.
2. ведомость при составляющих: дисциплина, студент, экзамен, семестр, преподаватель, оценка.
3. конференция по коммерческим вопросам при составляющих: дата, место проведения, тема, цель выступающие.
4. получение оценки при составляющих: преподаватель, студент, оценка, получать.
5. использования изделия при составляющих: организация, разработка технологического решения, исследование «физического эффекта», методы создания изделия.
6. информационная структура БД в машиностроении при составляющих: физические эффекты, технические решения, изделия, объект поставки изделия, приборы и стенды, нормативы.
7. классификация продукта при составляющих: название, область применения, способ хранения, способ транспортировки.
8. аудитория (описание) при составляющих: вместимость, назначение, составляющие, местонахождение.
9. животный мир при составляющих: вид, тип, среда обитания, особенности поведения.

4. Контрольные вопросы

1. Что представляет из себя фрейм, его составные части?
2. Что такое слот и из каких частей он состоит?
3. Для чего служат имя фрейма и имя слота?
4. Для чего служат указатели наследования?
5. для чего служат указание типа данных, демон?
6. Для чего служат присоединенная процедура и значение слота?

Лабораторная работа № 1. Часть 3. Описание предметной области. Разработка базы фактов и правил интеллектуальной системы

Цель работы: Научиться строить модель предметной области, описывать решаемую задачу правилами продукционной системы и формализовать используемые знания.

1. Теоретическая часть

В данной работе мы рассмотрим построение базы знаний на основе сведений, полученных от эксперта. Процесс ее построение состоит из двух этапов:

- описание предметной области;
- выбор метода и модели представления знаний;

Инженер знаний должен корректно сформулировать задачу. В то же время он должен уметь распознать, что задача не структурирована, и в этом случае воздержаться от попыток ее формализовать или применить систематические методы решения. Главная цель начального этапа построения базы знаний - определить, как будет выглядеть описание предметной области на различных уровнях абстракции. Экспертная система включает базу знаний, которая создается путем формализации некоторой предметной области, а та в свою очередь является результатом абстрагирования определенных сущностей реального мира.

После того как предметная область выделена, инженер знаний должен ее формально опи-

сать. Для этого ему необходимо выбрать какой-либо способ представления знаний о ней (модель представления знаний). В. настоящее время отсутствует общий способ представления знаний, который бы годился для формализации предметных областей любой природы. Инженер знаний должен воспользоваться той моделью, с помощью которой можно лучше всего отобразить специфику предметной области. Когда будет создана общая теория представления знаний (если это вообще когда-нибудь произойдет), ее можно будет применять для формализации новых предметных областей без учета их особенностей.

Определение характера решаемых задач

Обратимся к примеру из медицинской практики. Предположим, что мы хотим построить экспертную систему, предназначенную для обработки результатов химического анализа крови, выполненного в лаборатории. Инженер знаний прежде всего обязан провести опрос эксперта и только потом приступать к построению системы. Эксперт, безусловно, должен быть специалистом в той области, в которой будет работать система. Первым делом необходимо определить целевое назначение системы. Какие, собственно 'задачи предстоит решать системе, основанной на знаниях? Цели разработки системы следует сформулировать точно, полно и непротиворечиво. Например, для диагностической системы это может быть получение ответов на такие вопросы:

1. Здоров ли пациент (исправна ли система)? Если нет, то какое именно у него заболевание? Если имеете» несколько заболеваний, то какое из них наиболее опасно?

2. Какие изменения в диете и рационе питания следует рекомендовать и, какие из них считаются особенно важными?

3. Какие лабораторные исследования необходимо провести дополнительно и, какие из них являются первоочередными?

4. Как нужно изменить образ жизни пациента или климатические условия, в которых он находится?

5. Нужно ли направить пациента для обследования к врачам-специалистам и если да, то к каким именно? Подумайте, на какие еще вопросы должна уметь отвечать наша диагностическая система?

После того как цель разработки системы определена, инженер знаний приступает к формулированию подцелей. Это поможет ему установить иерархическую структуру системы и разбить ее на модули. Введение тех или иных подцелей обусловливается наличием связей между отдельными фрагментами знаний. Проблема сводится к разбиению задачи на две или несколько подзадач меньшей сложности и последующему поиску их решений. При необходимости, полученные в результате разбиения подзадачи могут дробиться и дальше.

Выявление объектов предметной области

Следующим шагом построения базы знаний является выделение объектов предметной области, или в терминах теории систем установление границ системы. Как и формальная система, *совокупность* выделенных понятий должна быть точной, полной и непротиворечивой. Итак, какие конкретно лабораторные анализы необходимо провести? Следует ли обратиться к истории болезни пациента и если да, то какие данные в ней наиболее важны? Какие еще сведения о пациенте могут представлять интерес (например, отмечались ли раковые заболевания у родственников)? Нужно ли учитывать лекарства, которые больной принимал ранее, а также предыдущие назначения врачей? Играет ли какую-нибудь роль род занятий и образ жизни больного, климатические условия и режим питания? Какие симптомы у него наблюдаются (головные боли, жар и т.д.)?

Установление взаимосвязей между объектами

После выявления объектов предметной области необходимо установить, какие между ними имеются связи. Например, низкое содержание тиреотропного гормона в крови может свидетельствовать о повышенной активности поджелудочной железы, но может означать и нечто другое. Следует стремиться к выявлению как можно большего количества связей, в идеале - всех, которые существуют в предметной области.

Формализация знаний

Полученное качественное описание предметной области должно быть представлено средствами какого-либо формального языка, чтобы привести это описание к виду, позволяющему по-

местить его в базу знаний системы. Для решения этой задачи выбирается подходящая модель представления знаний, с помощью которой сведения о предметной области можно выразить формально.

Рассмотрим пример.

Подходящей задачей, при решении которой можно использовать продукционную модель, может быть задача, вытекающая из следующей ситуации: к директору крупной технической фирмы пришёл человек, желающий устроиться на работу. Директор располагает сведениями о его квалификации, о потребностях фирмы в специалистах и общем положении дел в фирме. Ему нужно решить, какую должность в фирме может занять посетитель.

Рассмотрим модель «Посетитель», выявим необходимые атрибуты для принятия решения о приеме на работу.

Объект: посетитель.

Атрибуты:

1. наличие ученого звания
2. стаж работы по специальности
3. посетитель сделал важное открытие
4. средний бал посетителя за время учебы

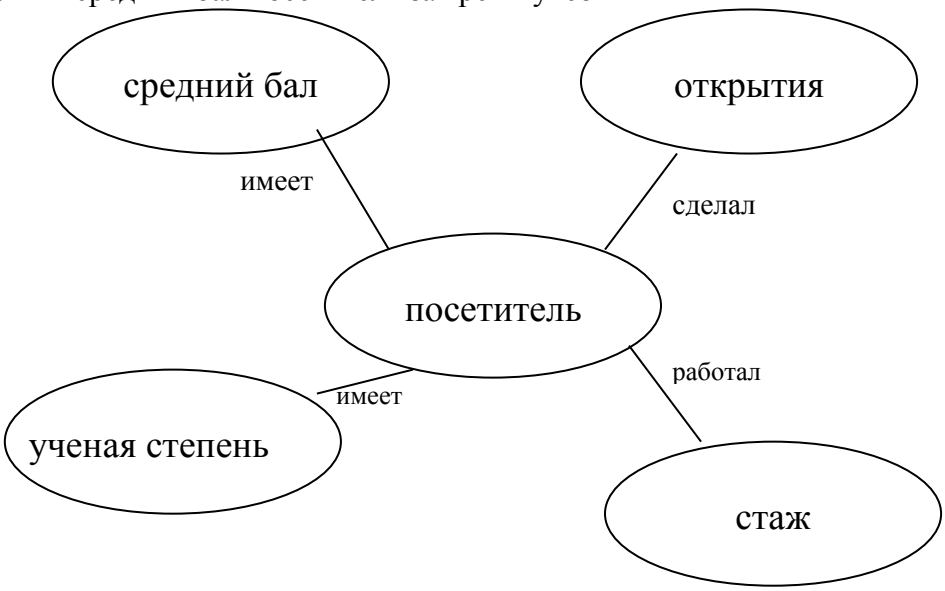


Рис.4. Модель предметной области

2. Порядок выполнения работы

1. Проанализировать полученное задание
2. Определить характер решаемой задачи.
3. Выделить объекты предметной области.
4. Выбрать атрибуты, свойства характеризующие объекты.
5. Установить связи между объектами в виде правил продукционной системы

3. Варианты заданий

Описать предметную область для следующих задач:

1. диагностика неисправностей электронной аппаратуры
2. диагностика неисправностей автомобиля
3. диагностика заболеваний (по выбору)
4. прогнозирование (по выбору)

a. спортивных мероприятий
b. телепередач
c. природных катаклизмов

и т.п.

5. классификация объектов (по выбору)
6. задачи информационно-советующего характера (по выбору)

a. помощник заведующего склада

b. помощник аптекаря
c. помощник оператора справочной службы
d. выбор должности
e. проведение отпуска
и т.п.

4. Контрольные вопросы

1. Какие модели представления знаний используются?
2. Типы задач экспертных систем?
3. Чем характеризуются объекты предметной области?
4. Как могут быть представлены факты в ЭС?

Лабораторная работа № 2. Часть 1. Использование правил продукции для представления знаний. прямая цепочка рассуждений

Цель работы: Научиться использовать метод правил продукции для представления знаний на основе прямой цепочки рассуждений.

1. Теоретическая часть

Представление знаний с помощью правил продукции – самая распространенная форма реализации БЗ. С помощью продукции можно описать практически любую систему знаний.

Правила продукций представлены в виде импликации:

$$p_i : s_i \rightarrow d_i ,$$

где p_i - правило продукции,

s_i - условие применения правила,

d_i - результат применения правила.

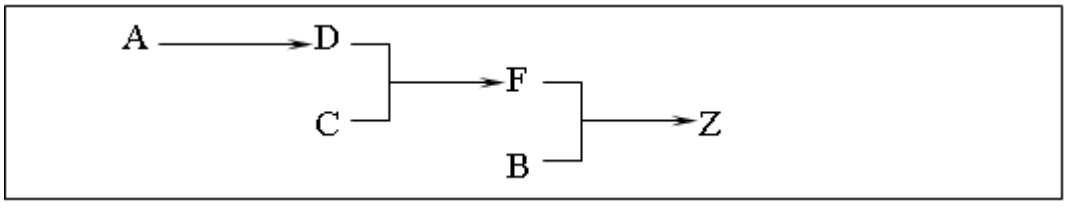


Рис.5. Пример использования правил продукции:

1. Если есть цены на выпускаемые изделия (A) - завод отпускает продукцию (D).
2. Если завод выпускает продукцию и выполняет план по ее реализации (C) - рабочие получают премию (F).
3. Если рабочие получают премию и растет производительность производства (B)- завод производит продукцию сверх плана (Z).

Рассмотрим цепочки выводов.

Прямой способ рассуждения.

По известным фактам отыскивается заключение, которое следует из этих фактов и накапливается рабочая память.

Это приводит к выполнению 2 правила.

C & D→ F, и факт «F» помещается в рабочую память. Тогда опять проверяются правила из базы. Первое правило выполняется $F \& B \rightarrow Z$, вследствие этого Z заносится в рабочую память. А так как Z является целью, то поиск заканчивается. Этот метод называется прямой цепочкой рассуждений, поскольку поиск новой информации происходит в направлении стрелок, разделяющих левые и правые части правил.

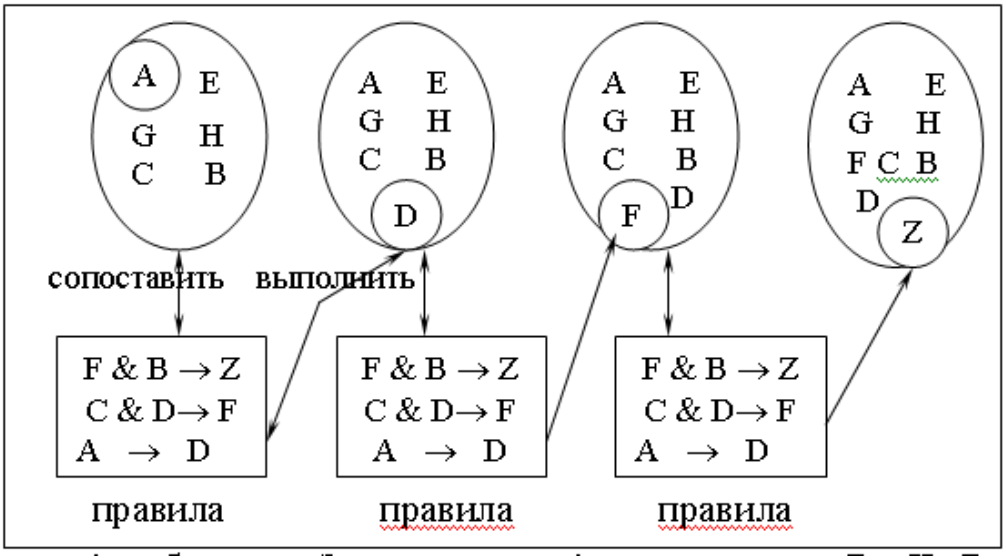


Рис.6. Пример реализации прямой цепочки рассуждений

Обобщённый алгоритм работы системы, реализующий прямую цепочку рассуждений, можно свести к следующему :

1. Определить исходное состояние.
2. Занести переменную условия в очередь переменных логического вывода, а её значение - в список переменных.
3. Просмотреть список переменных и найти ту переменную, имя которой стоит в начале очереди переменных логического вывода. Если переменная найдена, записать в указатель переменных условия номер правила и число 1. Если переменная не найдена, перейти к шагу 6.
4. Присвоить значения не проинициализированным переменным условной части найденного правила (если такие есть). Имена переменных содержатся в списке переменных условия. Проверить все условия правила и в случае их истинности обратиться к части ТО правила.
5. Присвоить значение переменной, входящей в часть ТО правила, и поместить её в конец очереди переменных логического вывода.
6. Удалить переменную, стоящую в начале очереди переменных логического вывода, если она больше не встречается в условной части какого-либо правила.

Закончить процесс рассуждений, как только опустеет очередь переменных логического вывода. Если же в очереди ещё есть переменные, вернуться к шагу 3.

2. Порядок выполнения работы:

1. Изучить теоретическую часть по приведенным выше данным и дополнительной литературе.
2. Просмотреть демонстрационный пример.
3. Получить у преподавателя вариант задания для выполнения.
4. Построить прямую цепочку рассуждений
5. Реализовать программу для прямой цепочки рассуждений

3. Варианты заданий

Реализовать прямую цепочку рассуждений для следующих задач:

1. прогнозирование неисправностей электронной аппаратуры
2. прогнозирование неисправностей автомобиля
3. прогнозирование заболеваний (по выбору)
4. прогнозирование (по выбору)

a. спортивных мероприятий
b. телепередач
c. природных катаклизмов

и т.п.

5. классификация объектов (по выбору)

6. задачи информационно-советующего характера (по выбору)
a. помощник заведующего склада
b. помощник аптекаря
c. помощник оператора справочной службы
d. выбор должности
e. проведение отпуска
и т.п.

4. Контрольные вопросы

1. Что такое правила продукции и в чем их сущность?
2. В чем отличие прямой цепочки рассуждений от обратной цепочки рассуждений?
3. Из каких частей состоит продукционная система?
4. Значение и применение частей продукционной системы для представления знаний?

Лабораторная работа № 2. Часть 2. Использование правил продукции для представления знаний обратная цепочка рассуждений

Цель работы: Научиться строить дерево целей и разрабатывать алгоритм Решений на основе обратной цепочки рассуждений.

1. Теоретическая часть

Прямой метод рассуждений имеет следующий недостаток. При большом количестве правил, чтобы найти информацию, связанную с Z, нужно выполнить много правил, не связанных с Z. При этом метод оказывается напрасной тратой времени и денег.

В таких ситуациях более рентабельной является обратная цепочка рассуждений.

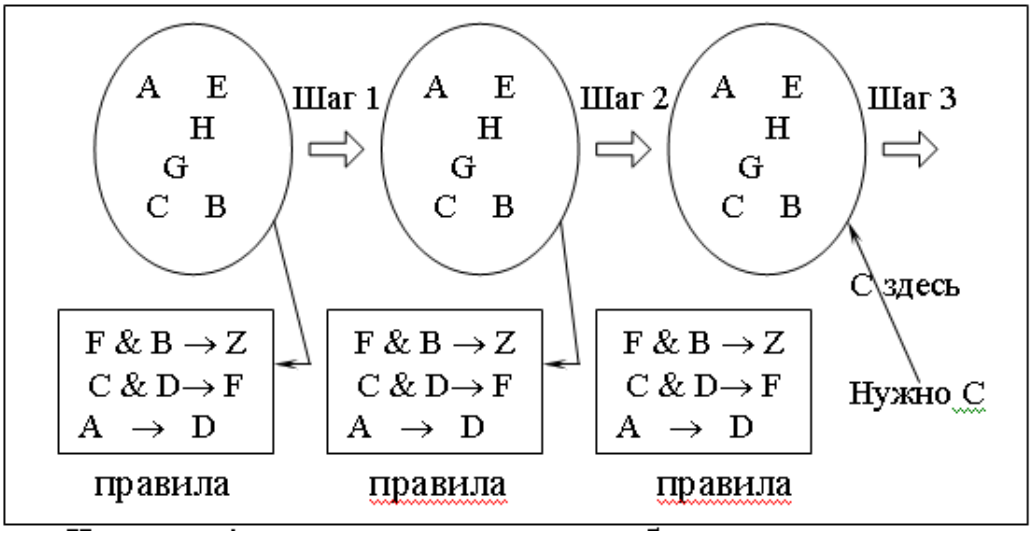


Рис.7 Пример реализации обратной цепочки рассуждений

При этом методе система начинает с того, что нужно доказать, например, что ситуация Z существует, и нужно выполнить только те правила, которые относятся к установлению этого факта.

На шаге 1 системе говорится, чтобы она установила, что ситуация Z существует. Она ищет Z в базе, а если его нет, будет искать правило, приводящее к установлению Z. Она находит правило

F & B → Z

и решает, что надо установить F и B.

На шаге 2 система пытается найти факт F или в базе данных или среди правил. Находит правило C & D→ F и решает, что необходимо установить существование фактов C и D.

На шагах 3-5 система находит C, затем находит A прежде, чем получит заключение о D.

На шагах 6-8 система выполняет третье правило, чтобы установить D, затем исполняет второе правило, чтобы установить F и наконец – первое правило, чтобы установить основную цель – факт существования Z.

Теперь нужно наглядно её представить. Для описания подобных задач обычно используются диаграммы, которые называются ***деревьями решений.*** Деревья решений дают необходимую наглядность и позволяют проследить ход рассуждений.

Диаграммы называются деревьями решений потому, что, подобно настоящему дереву, имеют ветви. Ветви деревьев решений заканчиваются логическими выводами. Для рассматриваемого примера вывод заключается в том, предложит ли директор должность поступающему на работу, и если да, то какую. Многие задачи сложны, и их непросто представить (или для их решения не собираются использовать ЭС). Дерево решений помогает преодолеть эти трудности.

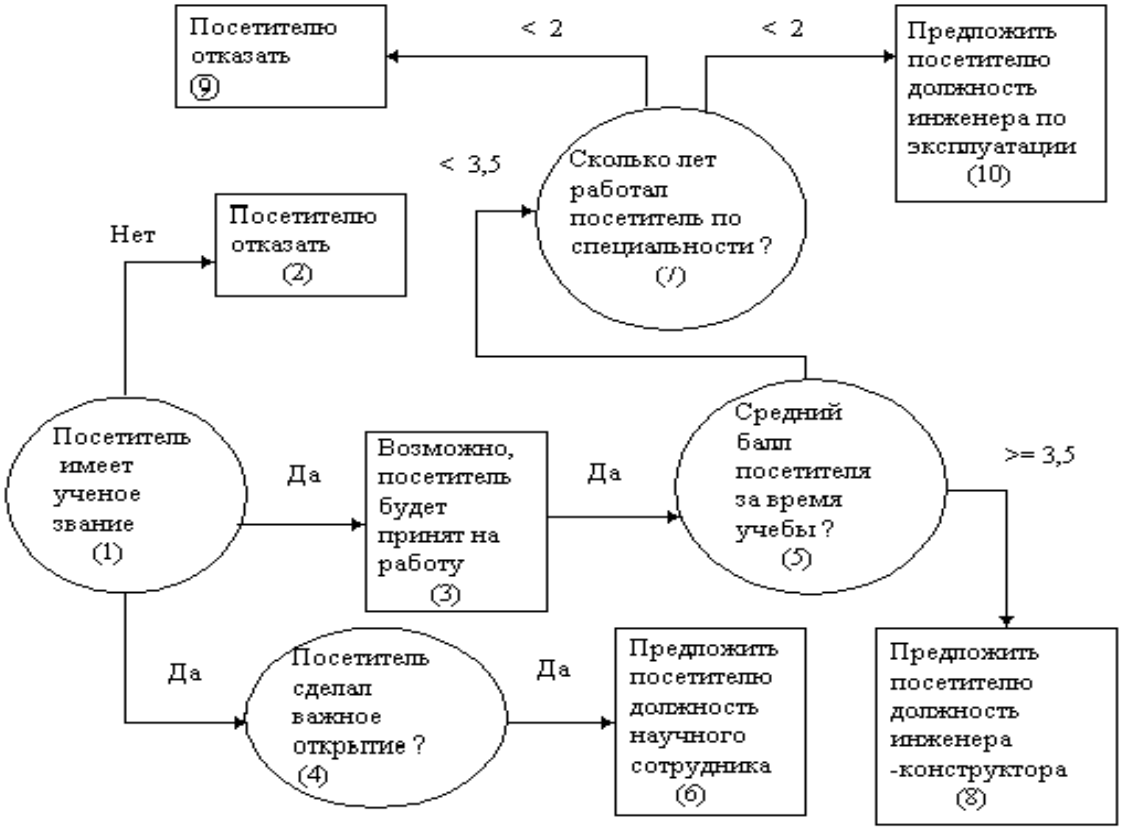


Рис. 8. Дерево решений для выбора должности.

На рис.8 показано дерево решений для примера с приёмом на работу. Видно, что диаграмма состоит из кружков и прямоугольников, которые называются вершинами. Каждой вершине присваивается номер. На вершины можно ссылаться по этим номерам. Линии, соединяющие вершины, называются дугами или ветвями. Кружки, содержащие вопросы, называются вершинами решений. Прямоугольники содержат цели диаграммы и означают логические выводы. Линии показывают направление диаграммы. Многие вершины имеют сразу по несколько ветвей, связывающих их с другими вершинами. Выбор выходящей из вершины ветви определяется проверкой условия, содержащегося в вершине.

Например, вершина 5 (см. Рис.8) содержит вопрос, на который есть два возможных ответа, и поэтому у неё два пути в зависимости от среднего балла посетителя за время учёбы, то есть возможен выбор одной из двух ветвей. Если средний балл равен 3.1, то будет выбран первый путь, так как 3.1 меньше 3.5. В программе под средний балл сначала отводится переменная, а затем ей присваивается значение. Можно сказать, что вершины содержат переменные, а пути - это условия, в соответствии с которыми переменным присваиваются значения. После того как для проблемной области сформулированы правила, эти условия становятся условными частями (ЕСЛИ) правила. Прямоугольники содержат частные или общие выводы. Например, прямоугольник на рис.1 может содержать ответ на вопрос, будет ли посетителю предложена работа. Общая цель системы, в кото-

рой реализованы обратные рассуждения, - получить окончательный ответ. Локальной целью может быть содержащийся в прямоугольнике на рис.8 ответ на вопрос, будет ли посетителю предложена должность. Однако эта вершина имеет и исходящие ветви, и, следовательно, через неё может проходить путь к следующему логическому выводу. В последнем случае, поскольку исходящая ветвь не содержит условия и она только одна, говорят, что вершина содержит локальный вывод для другой цели. Локальный вывод - это также составляющая условной части правила.

Обобщённый алгоритм работы системы с обратной цепочкой выводов.

Система, реализующая обратную цепочку рассуждений, должна выполнять следующие шаги :

1. Определить переменную логического вывода.
2. В списке логических выводов искать первое вхождение этой переменной. Если переменная найдена, в стек логических выводов поместить номер соответствующего правила и установить номер условия равным 1. Если переменная не найдена, сообщить пользователю, что ответ найти невозможно.
3. Присвоить значения всем переменным условия из данного правила.
4. Если в списке переменных указано, что какой-либо переменной условия не присвоено значение и её нет среди переменных логического вывода (её нет в списке логических выводов), запросить её значение у пользователя.
5. Если какая-либо переменная условия входит в переменные логического вывода, поместить в стек номер правила, в логический вывод которого она входит, и вернуться к шагу 3.
6. Если из правила нельзя определить значение переменной, удалить соответствующий ему элемент из стека и в списке логических выводов продолжить поиск правила с этой переменной логического вывода.
7. Если такое правило найдено, перейти к шагу 3.
8. Если переменная не найдена ни в одном из оставшихся правил в логическом выводе, правило для предыдущего вывода не верно. Если предыдущего вывода не существует, сообщить пользователю, что ответ получить невозможно. Если предыдущий вывод существует, вернуться к шагу 6.
9. Определить значение переменной из правила, расположенного в начале стека; правило из стека удалить. Если есть ещё переменные логического вывода, увеличить значение номера условия и для проверки оставшихся переменных вернуться к шагу 3. Если больше нет переменных логического вывода, сообщить пользователю окончательный вывод.

2. Порядок выполнения работы*:*

1. Выбрать по заданной теме 15-20 правил принятия решения;
2. Упорядочить их по степени важности;
3. Построить дерево принятия решения;
4. Построить список переменных логических выводов;
5. Написать программу для реализации обратной цепочки рассуждений

3. Варианты заданий

Реализовать прямую цепочку рассуждений для следующих задач:

1. диагностика неисправностей электронной аппаратуры
2. диагностика неисправностей автомобиля
3. диагностика заболеваний (по выбору)
4. Анализ объекта (по выбору)

a. спортивных мероприятий
b. телепередач
c. природных катаклизмов

и т.п.

5. задачи информационно-советующего характера (по выбору)

a. помощник заведующего склада
b. помощник аптекаря
c. помощник оператора справочной службы
d. выбор должности
e. проведение отпуска
и т.п.

4. Контрольные вопросы

1. Чем отличаются «прямая» и «обратная» цепочки рассуждений?
2. Какие виды правил существуют?
3. Как контролируется вывод правил из БЗ?
4. Как учитывается достоверность заключительной части правила?

Лабораторная работа № 3. Использование Simulink при построении нейронных сетей.

Целью работы является изучение основных блоков визуального приложения Simulink программной среде MATLAB 6.0, используемых при построении нейронных сетей, приобретение опыта моделирования нейросетевых систем управления среды MATLAB 6.0.

Теоретическая часть

Библиотека блоков построения нейронных сетей в Simulink.

Пакет Neural Network Toolbox содержит ряд блоков, которые могут быть непосредственно использованы для построения нейронных сетей в среде Simulink, либо применяться вместе с функцией **gensim** [1].

Для вызова набора блоков, в командной строке необходимо набрать команду **neural**, после выполнения которой появляется окно вида рис. 1.

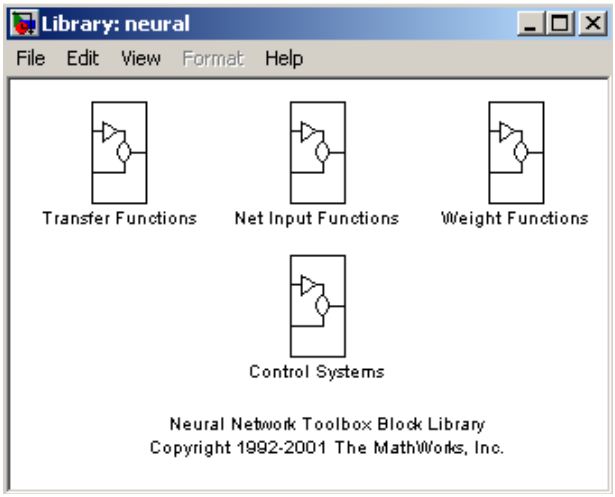


Рис. 1. Основные нейросетевые блоки Simulink

Каждый из представленных на рис. 1 блоков в свою очередь является набором (библиотекой) некоторых блоков. Рассмотрим их.

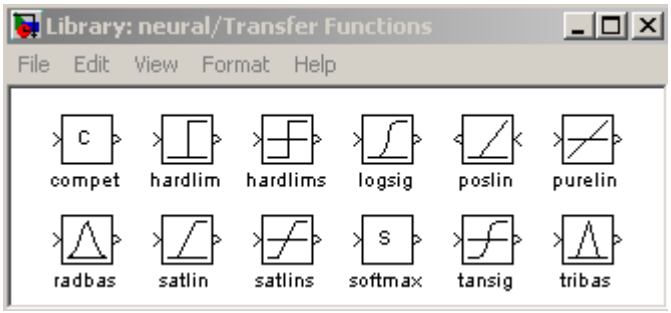


Рис. 2. Библиотека функций активации

Блоки функции активации (Transfer Functions). Двойной щелчок кнопки мыши на блоке Transfer Functions приводит к появлению библиотеки функций активации (рис. 2). Каждый из этих блоков данной библиотеки преобразует подаваемый на него вектор в соответствующий вектор той же размерности (табл. 1).

Таблица 1. Перечень функций активации нейронов

Название	Формула	Область значений
Пороговая	$f(x)=\begin{cases}0, & s<\theta,\\ 1, & s\geq\theta\end{cases}$	0,1
Знаковая (сигнатурная)	$f(x)=\begin{cases}1, & s>\theta,\\ -1, & s\leq\theta\end{cases}$	-1,1
Сигмоидальная (логистиничнская)	$f(x)=\frac{1}{1+e^{-s}}$	(0,1)
Полулинейная	$f(x)=\begin{cases}s, & s>\theta,\\ 0, & s\leq\theta\end{cases}$	(0,∞)
Линейная	$f(x)=s$	(-∞,∞)
Радиальная базисная (гауссова)	$f(x)=e^{-s^2}$	(0,1)
Полулинейная с насыщением	$f(x)=\begin{cases}0, & s\leq 0,\\ s, & 0<s<1,\\ 1, & s\geq 1\end{cases}$	(0,1)
Линейная с насыщением	$f(x)=\begin{cases}-1, & s\leq -1,\\ s, & -1<s<1,\\ 1, & s\geq 1\end{cases}$	(-1,1)
Гиперболический тангенс (сигмоидальная)	$f(x)=\frac{e^{s}-e^{-s}}{e^{s}+e^{-s}}$	(-1,1)

Треугольная	$f(x)=\begin{cases}1-\lvert s\rvert, & \lvert s\rvert\le\theta,\\ 1, & \lvert s\rvert>1\end{cases}$	(0,1)

Блоки преобразования входов сети. Проводя аналогичную рассмотренной операцию, но с блоком Net Input Functions, придём к библиотеке блоков вида, представленного на рис. 3 [1].

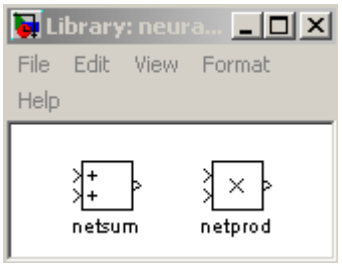


Рис. 3. Библиотека блоков преобразований сигналов

Блоки данной библиотеки реализуют рассмотренные выше функции преобразования входов сети.

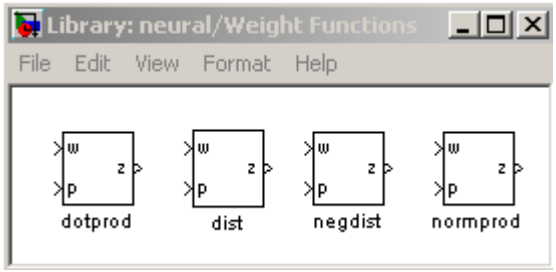


Рис. 4. Библиотека блоков весовых коэффициентов

Блоки весовых коэффициентов. Точно так же (но щелкая левой кнопкой мыши по иконке с надписью Weight Functions) придем к библиотеке блоков (рис. 4), реализующих некоторые функции весов и расстояний.

Отметим, что при задании конкретных числовых значений, при операции со всеми приведенными блоками ввиду особенностей Simulink векторы необходимо представлять как столбцы, а не как строки (как это было до сих пор).

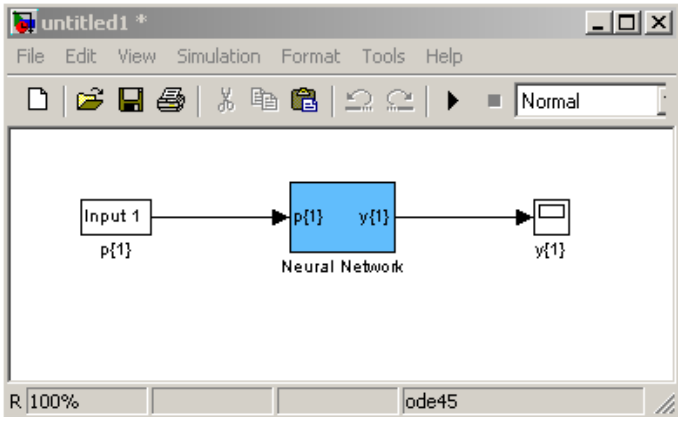


Рис. 5. Созданная нейросетевая модель Simulink

Формирование нейросетевых моделей. Основной функцией для формирование нейросетевых моделей в Simulink является функция **gensim,** записываемая в форме

gensim(net,st),

где **net –** имя созданной НС, **st –** интервал дискретизации (если НС не имеет задержек, ассоциированных с ее входами или слоями, значение данного аргумента устанавливается равным -1) [2].

Пример 1. Создание постейшей нейронной сети.

Пусть входной и целевой векторы имеют вид
p = [1 2 3 4 5];
t = [1 3 5 7 9].
Создадим линейную НС и протестируем ее:

```
» p = [1 2 3 4 5];
» t = [1 3 5 7 9];
» net = newlind(p,t);
» y =sim(net,p)
y = 1.0000    3.0000    5.0000    7.0000    9.0000
```

Затем запустим Simulink командой

```
» gensim(net,-1)
```

Это приведет к открытию блок-диаграммы (рис. 5).

Для проведения тестирования модели щелкнем дважды по левой иконке (с надписью Input 1, т. е. Вход 1), что приведет к открытию диалогового окна (рис. 6).

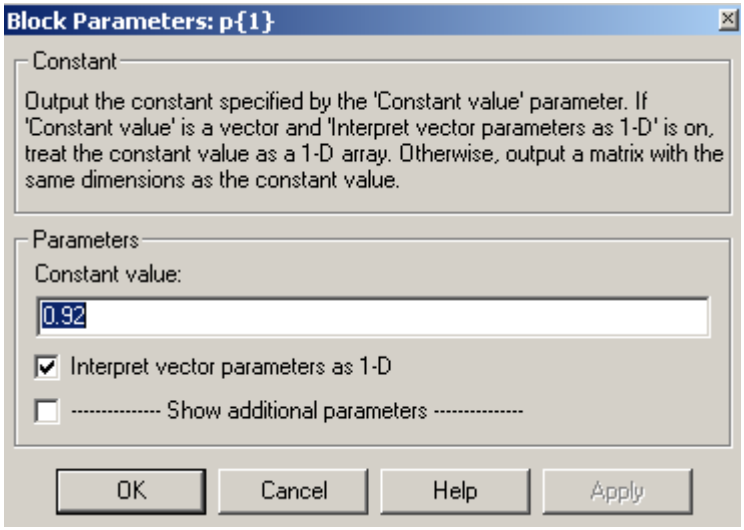


Рис. 6. Диалоговое окно задания входа НС

В данном случае блок Input 1 является стандартным блоком задания константы (Constant). Изменим значение по умолчанию на 2 и нажмем кнопку OK. Затем нажмем кнопку Start в меню моделирования. Расчет нового значения сетью производится практически мгновенно. Для его вывода необходимо дважды щелкнуть мышью по правой иконке (блоку y(1)). Результат вычислений отображается рис. 7 – он равен 3, как и требуется.

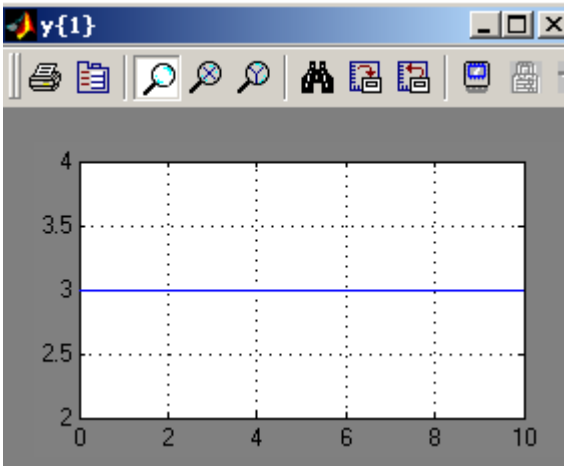


Рис. 7. Окно с выходом НС

Отметим, что, дважды щелкая левой кнопкой мыши по блоку Neural Network, затем – по блоку Layer 1, можно получить детальную графическую информацию о структуре сети (рис. 8).

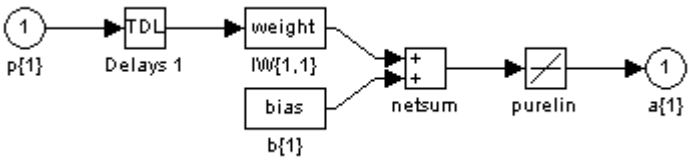


Рис. 8. Структура созданной НС

С созданной сетью можно проводить различные эксперименты, возможные в среде Simulink; вообще с помощью команды gensim осуществляется интеграция созданных нейросетей в блок-диаграммы этого пакета – с использованием имеющихся при этом инструментов моделирования различных систем (например, встраивание нейросетевого регулятора в систему управления и моделирование последней и т. п.).

Элементы математического моделирования неиросетевых систем управления в программной среде MATLAB 6.0

Библиотека инструментов пакета MATLAB 6.0 позволяет моделировать супервизорный режим работы многослойной нейросети в составе системы динамическим объектом, заданным уравнениями состояния. Реализация этого режима обеспечивается нейросетевым контроллером **Model Reference Control,** представляющим собой структуру, которая включает две многослойные нейронные сети: управляющую и идентификационную. Каждая сеть блока содержит один скрытый слой, количество нейронов в котором выбирается в зависимости от сложности задачи управления [2].

Блок **Model Reference Control** находится в библиотеке инструментов MATLAB в папке Neural Network Blockset/Control Systems.

Настройка контроллера осуществляется в два этапа. На первом этапе проводится идентификация объекта управления (обучение идентификационной многослойной нейронной сети (МНС)), а на втором – обучение управляющей МНС так, чтобы поведение замкнутой системы «контроллер – объект» управления было бы идентичным поведению заранее выбранной модели (reference-модели). Таким образом, в данном контроллере реализуется стратегия обучения «с учителем». В процессе обучения поэтапно производится настройка весовых коэффициентов нейронов сети по выбранному алгоритму. Под одним уроком обучения понимается перенастройка всех весовых коэффициентов сети.

При открытии блока **Model Reference Controller** появляется окно настройки управляющей сети. Для того чтобы перейти к идентификации ОУ, необходимо нажать на кнопку Plant Identification, расположенную в левом нижнем углу окна. При этом появится новое окно **Plant Identification** (рис. 9), в котором можно будет задать параметры, необходимые для настройки идентификационной сети.

Идентификация объекта управления происходит в три стадии.

Первая стадия – выбор структуры идентификационной МНС (блок **Network Architecture** на рис. 9). Она включает в себя выбор следующих параметров:

Size of Hidden Layer – количество нейронов в скрытом слое (используется МНС с одним скрытым слоем). Этот параметр следует выбирать с учетом сложности объекта управления. Следует учесть, что сеть с большим количеством нейронов в скрытом слое будет обеспечивать более высокие качественные показатели за меньшее количество уроков, но при этом возрастет сложность вычислений, т. е. один урок будет занимать большее время.

Sampling Interval – период дискретизации (в секундах).

No. Delayed Plan! Inputs – количество элементов задержки на входах идентификационной сети.

No. Delayed Plant Outputs – количество элементов задержки на выходах идентификационной сети.

Normalize Training Data – нормализация данных в процессе обучения. При выборе этого параметра данные, используемые в процессе обучения, будут приведены к отрезку [0; 1].

Вторая стадия – это получение данных для обучения идентификационной сети (блок **Training Data** на рис. 9). На этом этапе возможно импортирование данных из MAT-файла или

непосредственно из рабочего пространства MATLAB, а также формирование данных на основе заранее подготовленной модели объекта управления.

Модель объекта управления выполняется в виде отдельного файла SIMULINK и оформляется с помощью выделенных входа и выхода объекта (использовать блоки **Input Port** и **Output Port).**

В процессе формирования данных на основе представленной модели объекта управления проводится эксперимент, в котором на вход этой модели подается последовательность прямоугольных импульсов (в дальнейшем будем называть ее тестовым сигналом), параметры которой задаются следующими значениями:

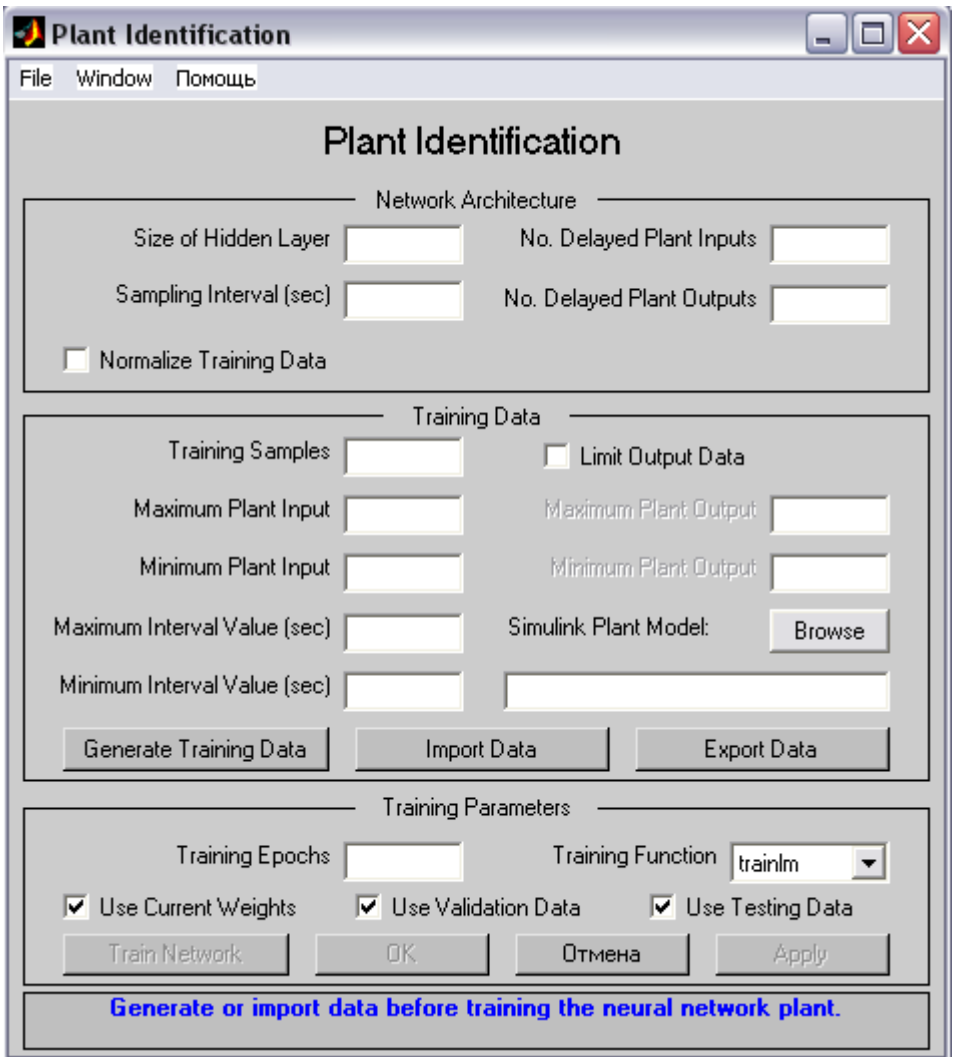


Рис. 9. Настройка процесса идентификации объекта управления

Training Samples – количество дискретных отсчетов в тестовой последовательности. Значение этого параметра следует выбирать исходя из сложности объекта управления.

Maximum Plant Input – максимальная величина сигнала.

Minimum Plant Input – минимальная величина сигнала.

Maximum Interval Value – максимальная длительность одного импульса (в секундах).

Minimum Interval Value – минимальная длительность одного импульса (в секундах).

Эти значения следует выбирать такими, чтобы тестовый сигнал предоставлял бы как можно большую информацию об объекте, т. е. был бы достаточно разнообразным.

Limit Output Data – ограничение выхода модели. При использовании этого параметра будет предложено выбрать минимальное и максимальное значения, которыми будет ограничена реакция объекта управления на тестовый сигнал.

Simulink Plant Model – здесь необходимо указать название модели SIMULINK. При нажатии на кнопку Browse будет предложено указать расположение заранее подготовленного файла с мо-

делью (для корректной работы путь, по которому находится файл, не должен содержать русских символов).

После выбора всех значений необходимо нажать кнопку Generate Training Data. При этом процесс выработка последовательности будет отображен в графическом окне **Plant Input-Output Data.** Процесс можно прервать, нажав на кнопку Stop Simulation.

После окончания будет предложено либо принять данные (Accept Data), либо отклонить их (Reject Data).

Если по каким-то причинам полученные данные не подходят, то при нажатии на кнопку Reject Data окно закроется, и можно будет задать другие параметры тестовой последовательности, другую модель или импортировать данные из MAT-файла или из рабочего пространства MATLAB.

При принятии данных кнопка Generate Training Data заменяется на Erase Generated Data. Таким образом, на любом этапе можно остановиться и сформировать тестовую последовательность заново.

Кнопка Import Data служит для импорта тестовых данных из сохраненного ранее MAT-файла или из рабочего пространства MATLAB. Кнопка Export Data предназначена для экспорта тестовых данных в MAT-файл или в рабочее пространство MATLAB в виде структуры данных или массива.

После получения тестовой последовательности необходимо перейти к заключительной стадии – обучению идентификационной МНС.

Для этого необходимо определить следующие параметры:

Training Epochs – количество уроков обучения. Значение этого параметра необходимо выбирать достаточно большим, чтобы ошибка обучения была бы достаточно мала. Однако, начинав с некоторого момента времени, дальнейшее обучение сети становится практически бесполезным, и поэтому количество уроков не должно быть слишком большим. Для данного примера обучение можно прекратить, когда график ошибки выйдет на длинный пологий участок.

Training Function – выбор алгоритма обучения. Для данного примера рекомендуется использовать алгоритм trainlm.

Use Current Weights – использовать уже полученные весовые коэффициенты. Этот параметр можно выбирать при повторном обучении сети.

Use Validation Testing Data – при выборе этих параметров 25% тестовых данных будет использовано для раннего завершения обучения и/или проверки сети. В нижней части окна синим цветом отображается подсказка.

После выбора всех параметров необходимо нажать на кнопку Train Network (обучение сети). При этом появится новое окно, в котором будет наглядно отображаться ход обучения, т. е. будет показана ошибка обучения для каждого урока. Процесс обучения можно прервать, нажав на кнопку Stop Training.

После завершения процесса обучения будет показано два новых окна – **Training data for NN Model Reference Control** и **Testing data for NN Model Reference Control** (если был выбран параметр Use Testing Data).

После успешного обучения идентификационной сети необходимо перейти к обучению управляющей сети. Для этого в окне Plant Identification нужно нажать на кнопку OK.

Для настройки процесса обучения управляющей сети предназначено окно **Model Reference Control (**рис. 10**).**

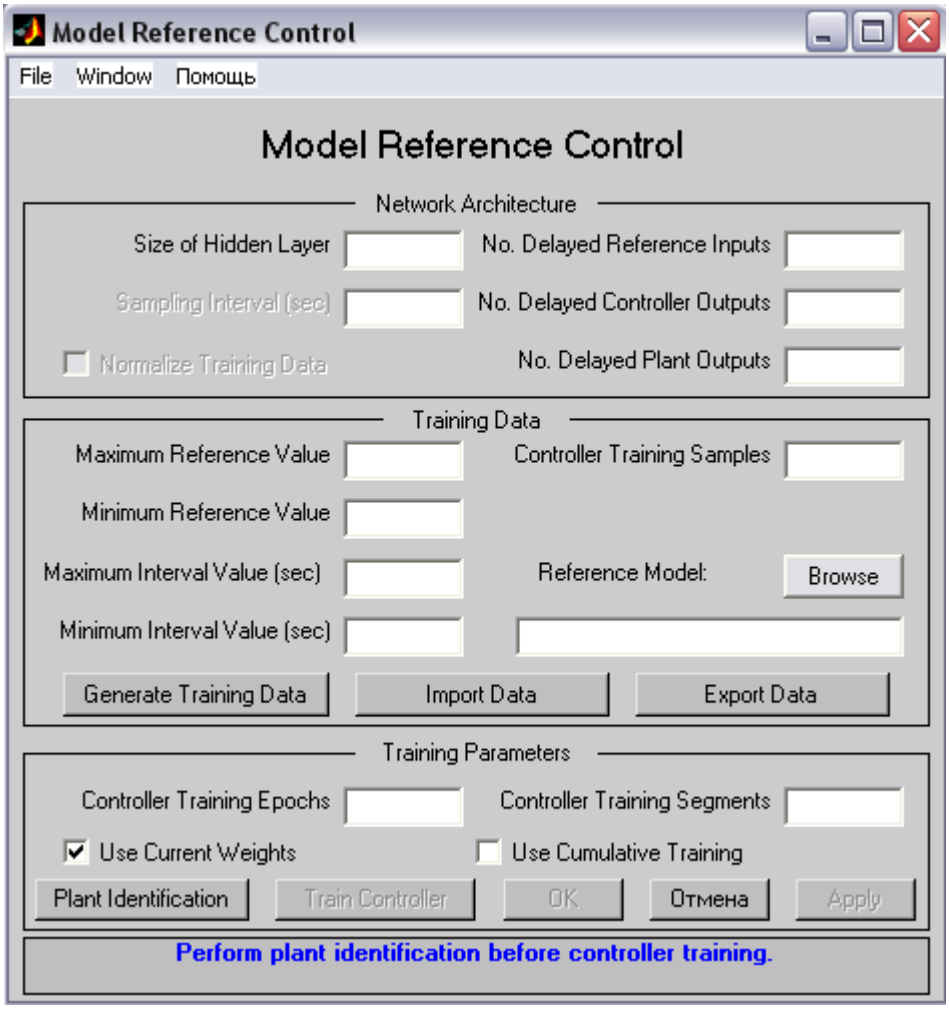


Рис. 10. Настройка процесса обучения управляющей сети

Процесс обучения контроллера также можно разбить на три стадии.

Первая стадия – выбор конфигурации управляющей сети

Здесь параметры Sampling Interval и Normalize Training Data выбираются такими же, как и при обучении идентификационной сети автоматически.

No. Delayed Reference Inputs – количество элементов задержки на входе, на который подается тестовый сигнал.

No. Delayed Controller Outputs – количество элементов задержки на выходе управляющей сети.

No. Delayed Plant Outputs – количество элементов задержки на выходе идентификационной сети.

Вторая стадия – получение данных для обучения.

На этой стадии, как и при обучении идентификационной сети, возможно импортирование данных из MAT-файла или непосредственно из рабочего пространства MATLAB, а также формирование данных на основе заранее подготовленной эталонной (reference) модели.

Reference-модель – это модель, которая, описывает желаемое поведение замкнутой системы контроллер-объект управления. Она задается так же, как и модель объекта управления на этапе идентификации, т. е. в виде файла на графическом языке SIMULINK.

Controller Training Samples – количество дискретных отсчетов а тестовом сигнале. Оно должно быть, как и при идентификации объекта, достаточно большим.

Все параметры, как и сам процесс формирования тестовой последовательности, аналогичны используемым на этапе идентификации объекта управления.

Третья стадия – обучение контроллера (управляющей сети). Для обучения необходимо задать следующие параметры:

Controller Training Epochs – количество уроков. Значение этого параметра нельзя определить заранее, поэтому его необходимо подобрать в процессе обучения.

Controller Training Segments – количество сегментов, на которые будут разделены тестовые данные. При этом для каждого сегмента будет проведено указанное количество уроков. Желательно разбивать данные на сегменты таким образом, чтобы каждый сегмент содержал хотя бы один переходный процесс.

Use Current Weights – использовать текущие весовые коэффициенты. В противном случае весовые коэффициенты будут выбраны случайным образом. Этот параметр следует применять при повторном обучении.

Use Cumulative Training – при выборе этого параметра первоначальное обучение будет выполнено с первым сегментом тестовых данных, а при последующем обучении остальные сегменты будут добавляться к данным, используемым на предыдущем этапе обучения. Таким образом, на финальной стадии обучения будет использован весь набор тестовых данных. Этот параметр следует использовать с осторожностью, так как он увеличивает время обучения.

После задания всех параметров необходимо приступить к обучению управляющей сети, для чего нужно нажать на кнопку Train Controller. При этом откроется окно, в котором будет наглядно отображаться процесс обучения (так же, как и при обучении идентификационной сети). Процесс обучения можно прервать, нажав на кнопку Stop Training.

Внимание! Процесс обучения может занимать длительное время (до нескольких часов), и при этом программа может не реагировать на нажатие кнопки Stop Training.

После завершения обучения открывается окно **Plant** Response **for** NN Model **Reference Control,** в котором отображаются результаты обучения. Их можно просмотреть более детально, выбрав в меню пункт View/Figure Toolbar. При этом под строкой меню появится панель инструментов, с помощью которой можно манипулировать изображением.

После успешного завершения процесса обучения управляющей сети контроллер готов к использованию в составе системы управления.

Возможные проблемы в процессе моделирования.

1. Процесс обучения идентификационной сети идет медленно или прекращается самопроизвольно, что определяется по тому, что ошибка обучения не уменьшается с какого-либо урока.

Это может происходить в следующих случаях:

- неверно выбрана конфигурация МНС2 – необходимо задать большее число нейронов в скрытом слое;
- неверно задан набор тестовых данных, не «возбуждающих» ОУ, – следует выбрать другие значения параметров в блоке **training data;**
- выбрано слишком малое число уроков обучения и его необходимо увеличить.

2. Процесс обучения управляющего контроллера (сети МНС1) идет крайне медленно и ошибка обучения практически не уменьшается.

Это может происходить в следующих случаях:

- неудачно заданы параметры тестовой последовательности при идентификации объекта и необходимо повторить процесс обучения сети МНС2 с другими параметрами в блоке training data; следует использовать близкие по форме и параметрам тестовые сигналы при обучении обеих МНС;
- выбрано слишком малое число элементов в скрытом слое и его необходимо увеличить.

3. Прямые показатели качества моделируемой системы существенно отличаются от аналогичных показателей для эталонной модели.

Это возможно в следующих случаях:

- идентификация объекта произведена неудачно, поэтому следует повторить настройку МНС2, учитывая ранее допущенные ошибки эксперимента, например, выбор недостаточного числа уроков обучения МНС2, числа нейронов в скрытом слое сети;

• конфигурация управляющей сети MHC1 не позволяет решать возложенную на нее задачу управления и необходимо повторить обучение с увеличенным количеством элементов в скрытом слое [2].

Пример 2. Система автоматического управления с нейросетевым регулятором на основе эталонной модели [3].

В качестве примера моделирования в среде Simulink систем управления с использованием нейросетевых регуляторов используем демонстрационный пример пакета, иллюстрирующий систему управления с эталонной моделью звеном («рукой») робота и вызываемый с помощью пунктов Help/Demos меню MATLAB (см. рис. 11).

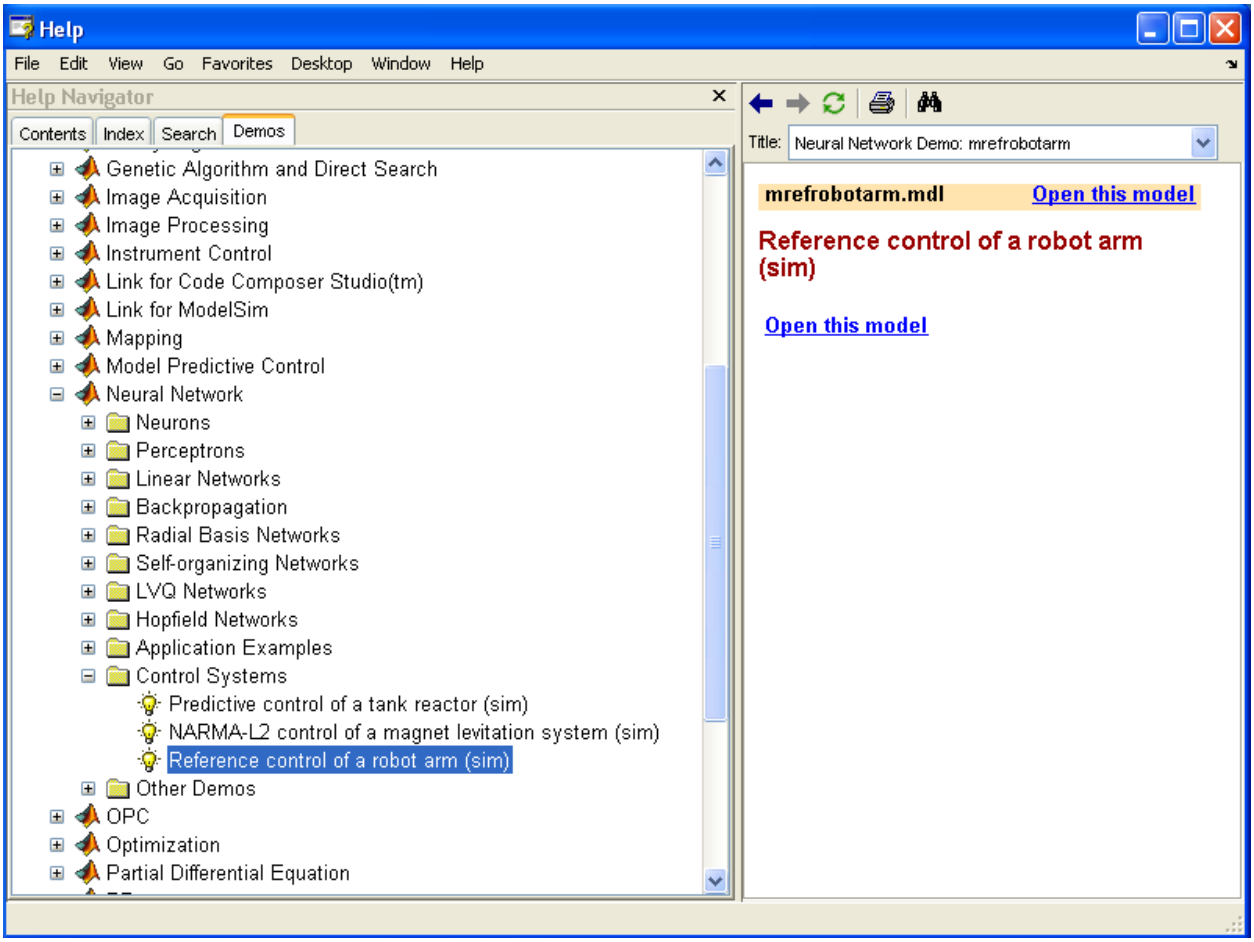


Рис. 11. Окно справки MATLAB

При открытии модели (с помощью мышки пункта Open this model (Отрыть эту модель)) в правой части окна, открывается окно Simulink с названием **mrefrobotarm** (рис. 12).

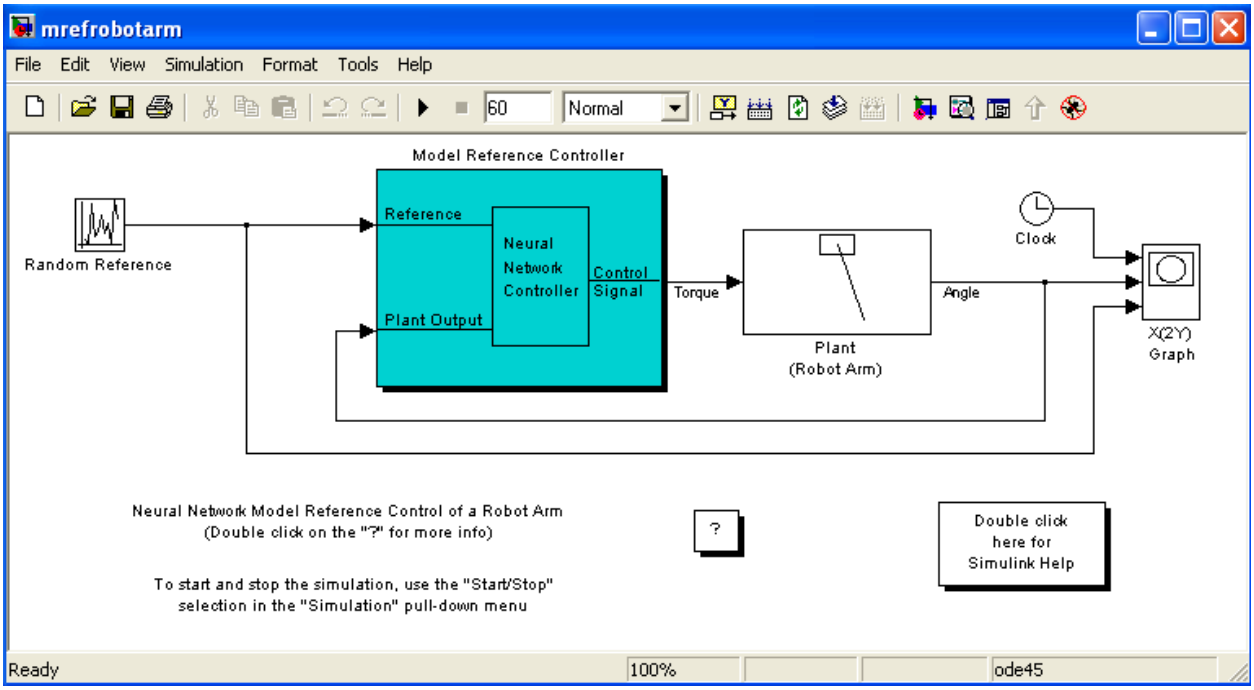


Рис. 12. Структура нейросетевой системы управления в среде Simulink

При реализации системы использованы 2 нейронные сети – для регулятора и для модели объекта управления.

В этом демонстрационном примере цель заключается в управлении движением одного звена робота, схематично показанного на рис. 13.

Рис. 13. Схематичное изображение движения звена робота

Управление движения звена представлено соотношением:

$$\frac{d^2\phi}{dt^2} = -10\sin\phi - 2\frac{d\phi}{dt} + u,$$

где φ – угол поворота звена, u – момент, развиваемый двигателем постоянного тока.

Соответствующая динамическая модель, реализаванная в среде Simulink, приведена на рис. 14.

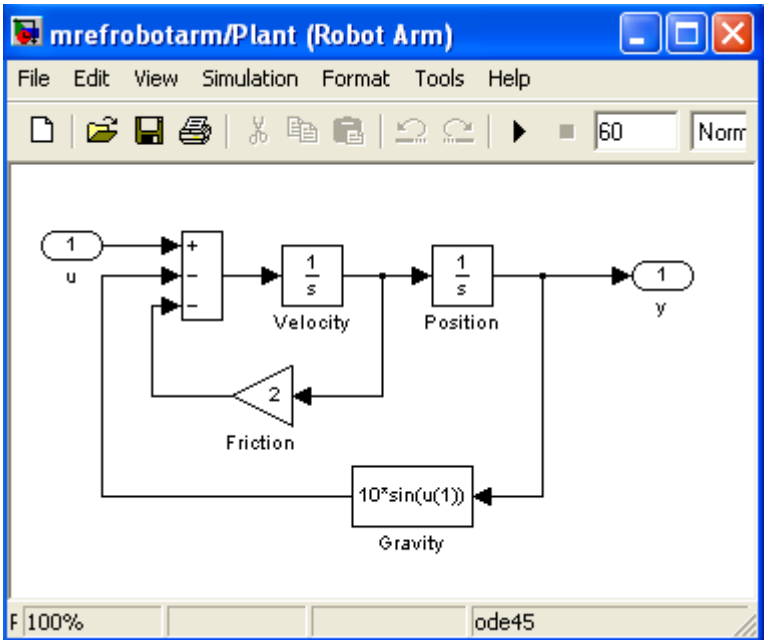


Рис. 14. Структурная динамическая модель звена робота

Цель обучения регулятора состоит в том, чтобы движение звена соответствовало выходу эталонной модели:

$$\frac{d^2 y_r}{dt^2} = -9\sin y_r - 6\frac{dy_r}{dt} + 9r\,,$$

где y_r – выход эталонной модели, r – задающий сигнал на входе модели.

Структурная схема, поясняющая принцип построения системы управления с эталонной моделью, показана на рис. 15.

Рис. 15. Система управления с эталонной моделью

На рисунке:
Command Input – задающий сигнал,
Reference Model – эталонная модель,
NN Controller – нейросетевой регулятор,
NN Plant Model – нейронная сеть модели объекта,
Plant – объект управления,
Model Error – ошибка модели,
Control Error – ошибка управления,
Plant Output – выход объекта.

В приведенной структуре следует выделить эталонную модель, которая задает желаемую траекторию движения звена робота, удовлетворяющему второму из приведенных дифференциальных уравнений, а также нейронные сети, реализующие регулятор и объект управления.

В данном случае архитектура НС регулятора описывается профилем 5–13–1 (5входов, 13 нейронов скрытого слоя и один выход), т. е. содержит два слоя нейронов. Из двух слоев состоит и нейронная сеть модели объекта. Линии задержки, используемые для формирования входов НС, имеют такт дискретности 0.05 с.

Для начала работы с демонстрационным примером необходимо активизировать блок **Model Reference Controller** (см. рис. 12) двойным щелчком левой кнопки мыши. Появится окно, показанное на рис. 16.

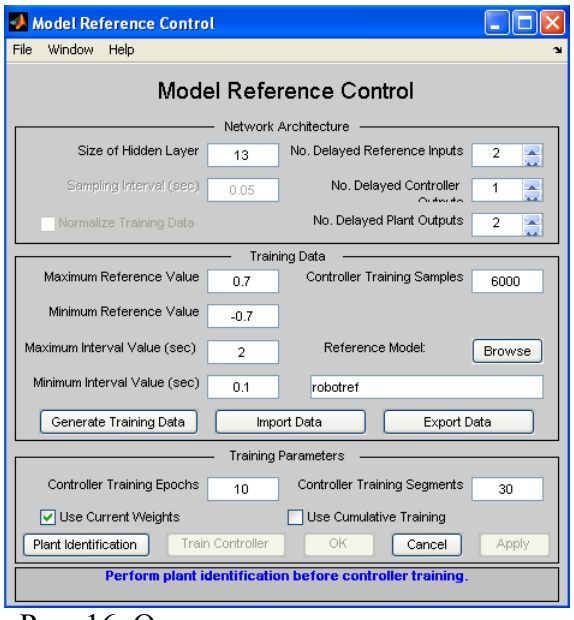


Рис. 16. Окно настроек нейросетевых моделей

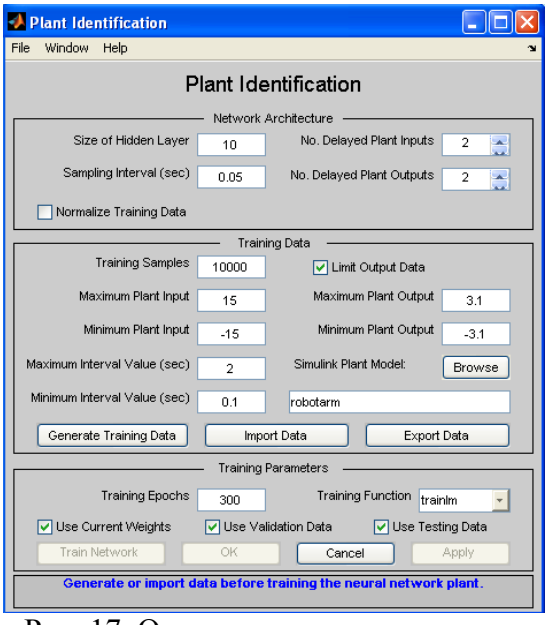


Рис. 17. Окно для построения нейросетевой модели ОУ

Особенность рассматриваемой системы управления состоит в том, что следует построить две нейронные сети – модель объекта управления и самого регулятора. Начнем с построения модели объекта управления, выбрав в окне настроек кнопку Plant Identification. При этом откроется окно **Plant Identification**, которое показано на рис. 17.

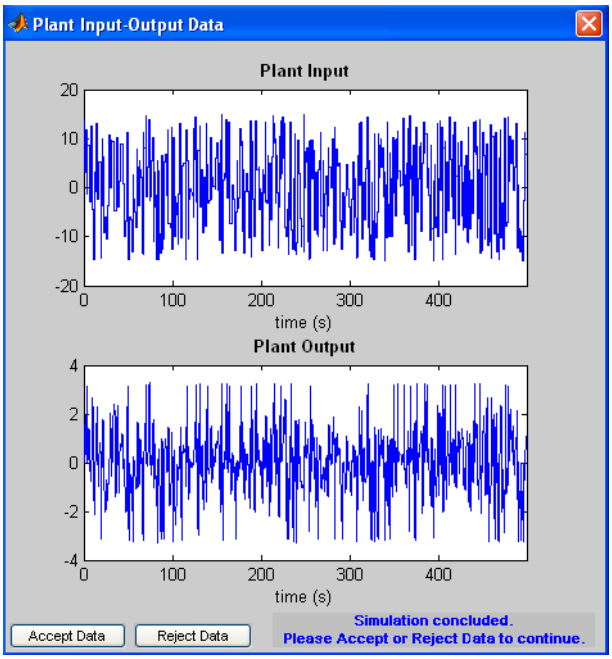


Рис. 18. Сигнал ОУ в режиме его идентификации

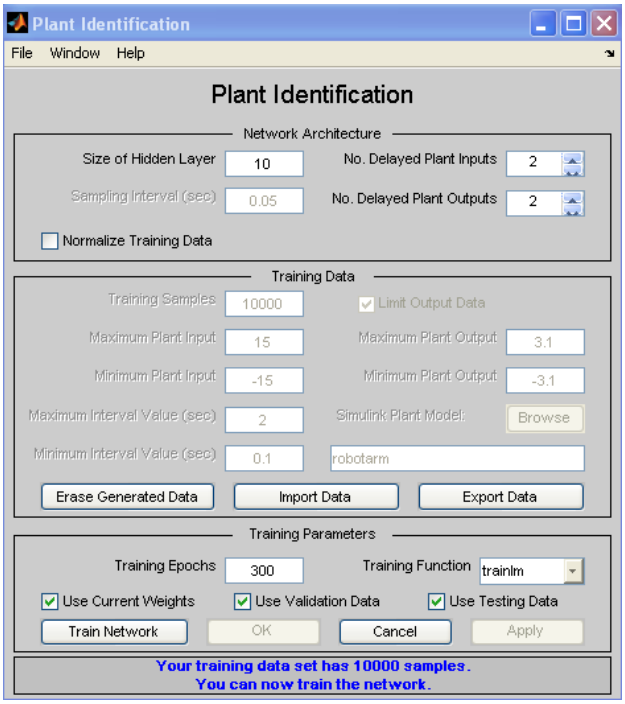


Рис. 19. Модифицированное окно для построения модели ОУ

Для построения нейросетевой модели ОУ следует выбрать кнопку Generate Training Data. Это приведет к тому, что будет запущена программа генерации обучающей последовательности на интервале 500 с для модели рассматриваемого объекта управления. Программа генерирует обучающие данные путем воздействия ряда случайных ступенчатых сигналов на модель Simulink объекта. Графики входного и выходного сигнала объекта выводятся на экран (рис. 18).

По завершении генерации обучающей последовательности пользователю предлагается либо принять данные (Accept Data), либо отказаться от них (Reject Data).

Примем данные, после чего приложение возвратит нас к несколько измененному окну **Plant Identification** (рис. 19). Здесь часть окна недоступна, а кнопка Generate Training Data заменена на кнопку Erase Generated Data, что позволяет удалить сгенерированные данные.

Теперь можно начать обучение нейронной сети.

Для этого следует воспользоваться кнопкой Train Network (Обучить сеть). Начнется обучение нейросетевой модели. После завершения обучения его результаты отображаются на графиках, как это показано на рис. 20 и 21, где построены соответствующие результаты обучения и тестирования на контрольном множестве.

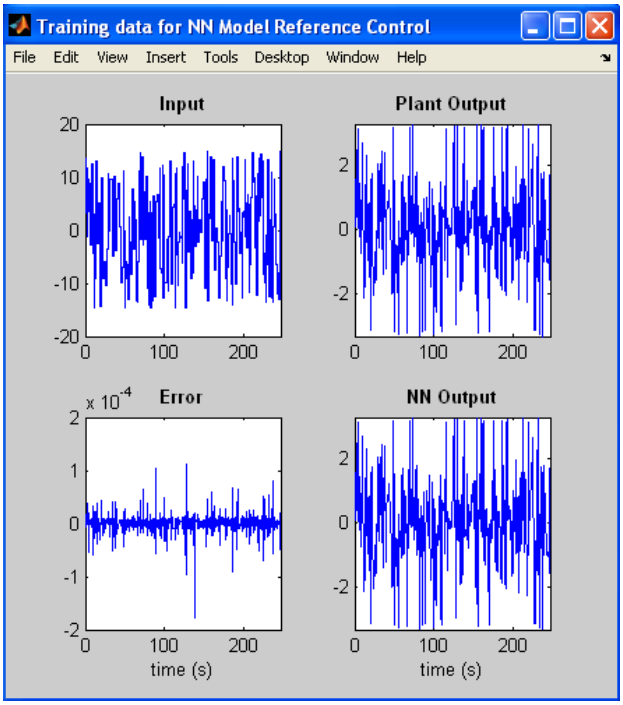


Рис. 20. Результат обучения модели на обучающем множестве

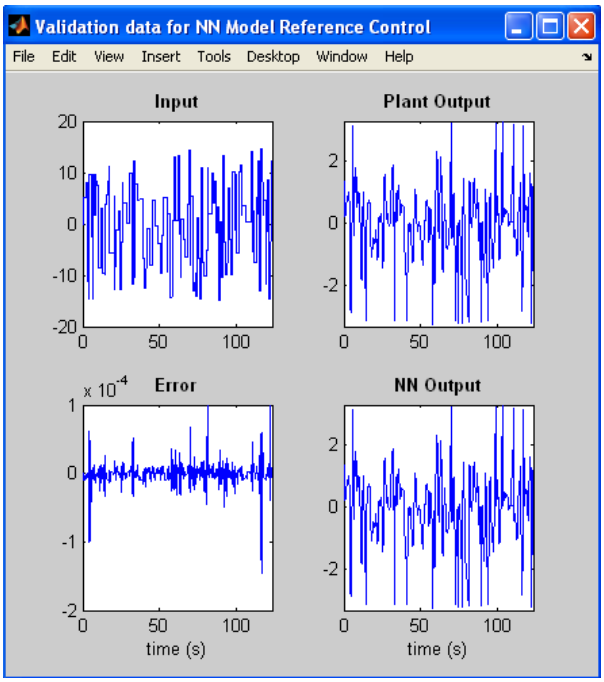


Рис. 21. Результат тестирования модели на контрольном множестве

Текущее состояние отмечено в окне **Plant Identification** (рис. 22) сообщением «Обучение завершено». Вы можете сгенерировать или импортировать новые данные, продолжить обучение или сохранить полученные результаты, выбрав кнопки OK или Apply.

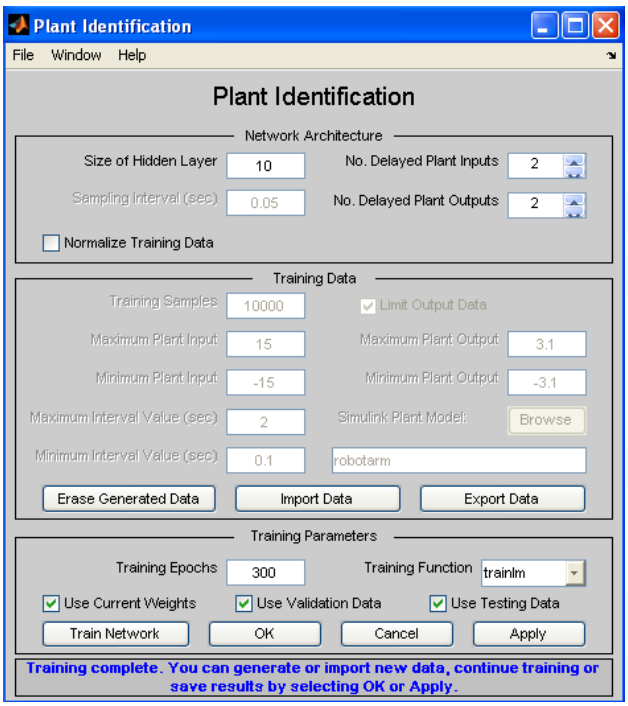


Рис. 22. Окно идентификации объекта с сообщением об окончании процедуры

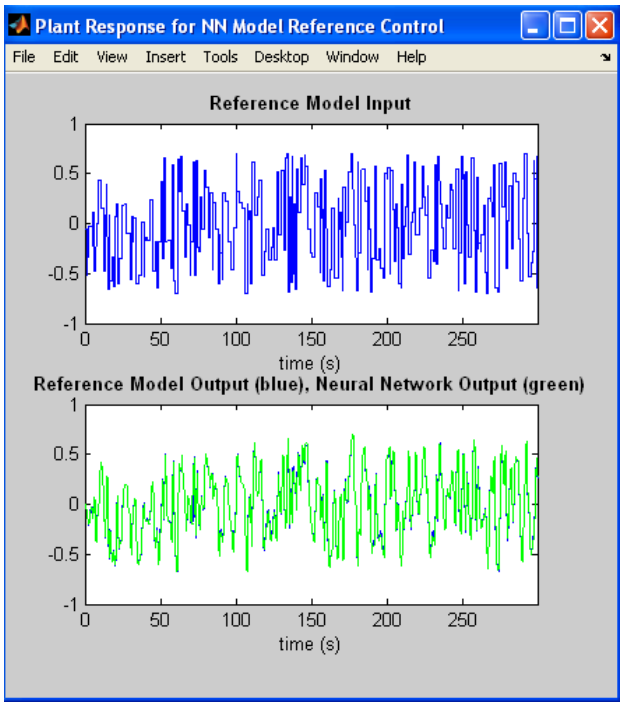


Рис. 23. Выход эталонной модели и объекта управления

Выберем кнопку OK. В результате параметры нейросетевой модели объекта управления будут введены в блок **Model Reference Controller** системы Simulik, после чего произойдет возврат в окно **Model Reference Controller** (рис. 23).

Теперь для обучения регулятора следует выбрать кнопку Generate Training Data, чтобы сгенерировать обучающие данные. Эти данные в виде графиков появятся в окне **Input-Output Data for NN Model Reference Controller** и снова необходимо подтвердить или опровергнуть эти данные. Если данные приемлемы, то в окне **Model Reference Controller** следует выбрать кнопку Train

Controller (Обучить регулятор). После того как обучение закончится, графики выходов эталонной модели и объекта управления выводятся на экран (рис. 24). Заметим, что обучение регулятора занимает, в силу применяемого алгоритма обучения (динамического варианта метода обратного распространения ошибки), весьма значительное время.

По окончании обучения регулятора необходимо нажать на кнопку OK, вернуться к модели Simulik (рис. 12) и начать моделирование, выбрав опцию Start из меню Simulation. Графики эталонного сигнала и выхода объекта управления показаны на рис. 25.

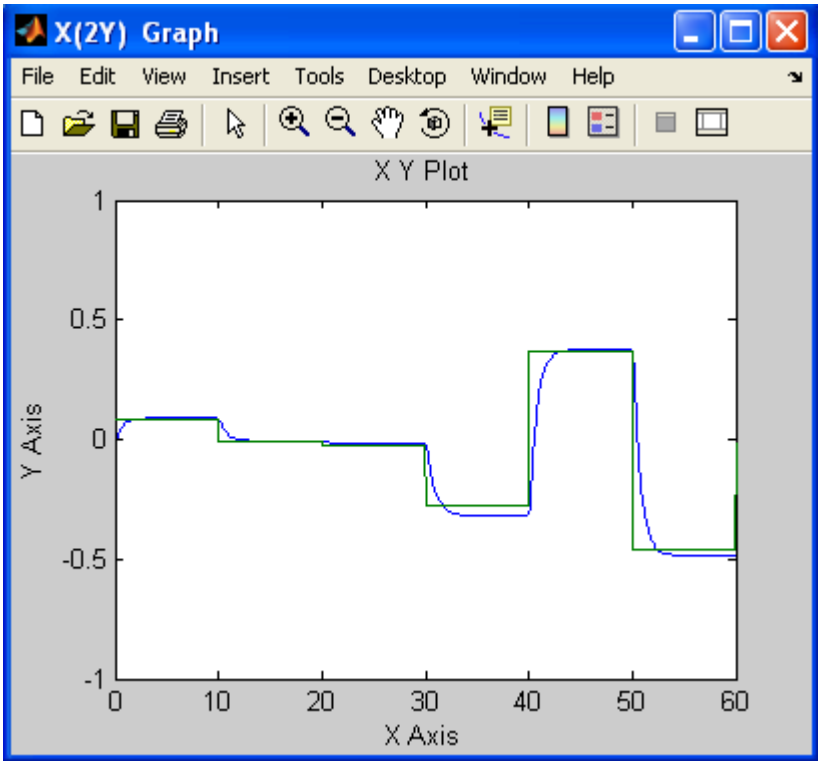


Рис. 25. Графики эталонного сигнала (ступенчатый процесс) и выхода объекта

Из анализа полученных данных следует, что реакция системы на ступенчатые воздействия со случайной амплитудой носит монотонный характер и обрабатывается в пределах 3…4 с. Это свидетельствует о хорошем качестве регулятора Model Reference Controller для управления звеном робота-манипулятора.

Программа работы и методические указания

1) Выполните все примеры, предлагаемые в теоретической части.
2) Для примера 2 исследуйте в моделируемой схеме нейросетевой адаптивной системы влияние параметров нейросети и числа уроков обучения на показатели качества процесса идентификации объекта (первый этап адаптивного управления) и процесса обучения сети MHC1 (второй этап адаптивного управления).
3) Отобразите результаты обучения (для примера 2) для различного числа нейронов в скрытом слое.

Лабораторная работа № 4. Графический интерфейс гибридных систем

Цель работы. Создать гибридную систему по ранее выполненной зависимости $y = x^2$.

Графический интерфейс гибридных (нечетких) нейронных систем вызывается функцией ***anfisedit*** (из режима командной строки). Использование функции приводит к появлению окна редактора гибридных систем (ANFIS Editor, ANFIS-редактор).

С помощью данного редактора осуществляются создание или загрузка структуры гибридной системы, просмотр структуры, настройка ее параметров, проверка качества функционирования такой системы. Создание структуры, настройка параметров и проверка осуществляются по выборкам (наборам данных) — обучающей (Training data), проверочной (Checking data) и тестирующей (Testing data). Предварительно они должны быть представлены в виде текстовых файлов (с расширением dat и разделителями-табуляциями), первые столбцы которых соответствуют входным переменным, а последний (правый) — единственной выходной переменной; количество строк в таких файлах равно количеству образцов (примеров). Так, обучающая выборка, сформированная по табл. 1, представляется в виде Proba. dat

Строгих рекомендаций по объемам указанных выборок не существует; по-видимому, лучше всего исходить из принципа «чем больше, тем лучше».

Обучающая и проверочная выборки непосредственно задействуются в процессе настройки параметров гибридной сети. Проверочная — для выяснения ситуации, не происходит ли так называемого *переобучения* сети, при котором ошибка для обучающей последовательности стремится к нулю. Для проверочной — возрастает; впрочем, наличие проверочной выборки не является строго необходимым, оно лишь крайне желательно. Тестовая (или тестирующая) выборка применяется для проверки качества функционирования настроенной (обученной) сети. Поясним пункты меню и опции редактора.

Меню **File** и **View** в общем идентичны аналогичным меню **FIS**-редактора за тем исключением, что здесь работа может происходить только с алгоритмом нечеткого вывода **Sugeno**. Меню **Edit** содержит единственную команду — **Undo** (Отменить выполненное действие). Набор опций **Load data** (Загрузка данных) в нижней левой части окна редактора включает в себя:

- тип (Type) загружаемых данных (для обучения — Training, для тестирования — Testing, для проверки — Checking, демонстрационные — Demo);
- место, откуда должны загружаться данные: с диска (disk) или из рабочей области MATLAB (workspace).

К данным опциям относятся две кнопки, нажатие на которые приводит к требуемым действиям **Load Data** (Загрузить данные) и **Clear Data** (Стереть данные).

Следующая группа опций (в середине нижней части окна **ANFIS**редактора) объединена под именем **Generate FIS** (Создание нечеткой системы вывода). Данная группа включает в себя следующие опции:

- загрузку структуры системы с диска (from disk);
- загрузку структуры системы из рабочей области MATLAB (from worksp.);
- разбиение (деление) областей определения входных переменных (аргументов) на подобласти — независимо для каждого аргумента (**Grid partition);**
- разбиение всей области определения аргументов (входных переменных) на подобласти — в комплексе для всех аргументов (**Sub.clustering**),

Кроме того, имеется также кнопка **Generate FIS**, нажатие которой приводит к процессу создания гибридной системы с точностью до ряда параметров.

Следующая группа опций — **Train FIS** (Обучение нечеткой системы вывода) — позволяет определить метод «обучения» (Optim.Method) системы (т. е. метод настройки ее параметров) — гибридный (hybrid) или обратного распространения ошибки (backproba), а также установить уровень текущей суммарной (по всем образцам) ошибки обучения (Error To1erance), при достижении которого процесс обучения заканчивается и количество циклов обучения (Epochs), т.е. количество «прогонов» всех образцов (или примеров) обучающей выборки. Процесс обучения, таким образом, заканчивается либо при достижении отмеченного уровня ошибки обучения, либо после проведения заданного количества циклов.

Кнопка **Train Now** (Начать обучение) запускает процесс обучения, т.е. процесс настройки параметров гибридной сети.

В правом верхнем углу окна ANFIS-редактора выдается информация (ANFIS Info) о проектируемой системе: количество входов, выходов, функций принадлежности входов. Нажатие кнопки **Structure** (Структура) позволяет увидеть структуру сети. Кнопка **C1ear** (Очистить) позволяет стереть все результаты.

Опции **Test FIS** в правом нижнем углу окна позволяют провести проверку и тестирование

созданной и обученной системы с выводом результатов в виде графиков. Соответствующие графики для обучающей выборки — **Training data**, тестирующей выборки — **Testing data** и проверочной выборки — **Checking data**. Кнопка **Test Now** позволяет запустить указанные процессы.

Задание к лабораторной работе 2

Работу с редактором рассмотрим на примере восстановления зависимости $y = x^2$ по данным табл. 1. Предположим, что эти данные подготовлены в файле Proba. dat. Создание и проверку системы, как и раньше, проведем по этапам:

1. В окне ANFIS-редактора выберем тип загружаемых данных Training и нажмем кнопку Load data. В последующем стандартном окне диалога укажем местоположение и имя файла. Его открытие приводит к появлению в графической части окна редактора набора точек, соответствующих введенным данным.

2. В группе опций **Generate FIS** по умолчанию активизирован вариант **Grid partition**. Не будем изменять и нажмем кнопку **Generate FIS**, после чего появится диалоговое окно для задания числа и типов функций принадлежности. Сохраним все установки по умолчанию, согласившись с ними нажатием кнопки ОК. Произойдет возврат в основное окно ANFIS редактора. Теперь структура гибридной сети создана, и ее графический вид можно просмотреть с помощью кнопки **Structure**.

3. Перейдем к опциям **Train FIS**. Не будем менять задаваемые по умолчанию метод настройки параметров (hybrid — гибридный) и уровень ошибки (0), но количество циклов обучения изменим на 40, после чего нажмем кнопку запуска процесса обучения (**Train Now**). Получился результат в виде графика ошибки сети в зависимости от числа проведенных циклов обучения (из которого следует, что фактически обучение закончилось после пятого цикла).

4. Теперь нажатием кнопки **Test Now** можно начать процесс тестирования обученной сети, но, поскольку использовалась только одна (обучающая) выборка, ничего особенно интересного ожидать не приходится. Действительно, выход обученной системы практически совпадает с точками обучающей выборки.

5. Сохраним разработанную систему на диске в файле с именем Proba1 (с расширением *fis*) и для исследования разработанной системы средствами FIS-редактора из командной строки MATLAB выполним команду **Fuzzy**, а затем через пункты меню **File | Open FIS from disk** откроем созданный файл. С созданной системой можно теперь выполнять все приемы редактирования (изменение имен переменных и т. п.) и исследования, которые были рассмотрены выше. В результате убеждаемся, что качество аппроксимации существенно не улучшилось – слишком мало данных. Что можно сказать про эффективность использования гибридных систем и ANFIS-редактора?

В данном случае используется только один алгоритм нечеткого вывода — **Sugeno** (нулевого или первого порядков), может быть только одна выходная переменная, всем правилам приписывается один и тот же единичный вес. Вообще говоря, возникают значительные проблемы при большом (более 5-6) количестве входных переменных. Это — ограничения и недостатки подхода.

Его несомненные достоинства: практически полная автоматизация процесса создания нечеткой (гибридной) системы, возможность просмотра сформированных правил и придания им содержательной (лингвистической) интерпретации, что позволяет, кстати говоря, рассматривать аппарат гибридных сетей как средство извлечения знаний из баз данных и существенно отличает данные сети от классических нейронных.

Рекомендуемая область применения: построение аппроксиматоров зависимостей по экспериментальным данным, построение систем классификации (в случае бинарной или дискретной выходной переменной), изучение механизма явлений.

Затем работу с редактором рассмотрим на примере зависимости *y* от *x*, заданной по вариантам.

Лабораторная работа № 5. Часть 1. Использование графического интерфейса для построения нечеткой аппроксимирующей системы

Цель работы. Построение нечеткой экспертной системы.

Рассмотрим методику построения нечеткой экспертной системы, которая должна помочь пользователю с ответом на вопрос: сколько дать на чай официанту за обслуживание в ресторане?

На основании каких-то устоявшихся обычаев и интуитивных представлений предположим, что задача о чаевых может быть описана следующими предложениями.

1. Если обслуживание плохое или еда подгоревшая, то чаевые – маленькие.
2. Если обслуживание хорошее, то чаевые – средние.
3. Если обслуживание отличное или еда превосходная, то чаевые щедрые.

Качество обслуживания и еды будем оценивать по 10-балльной системе (0 – наихудшая оценка, 10 наилучшая).

Будем предполагать далее, что малые чаевые составляют около 5% от стоимости обеда, средние – около 15 % и щедрые – примерно 25 %.

Заметим, что представленной информации, в принципе, достаточно для проектирования нечеткой экспертной системы. Такая система будет иметь 2 входа (которые можно условно назвать «сервис» и «еда»), один выход («чаевые»), три правила типа если….то (в соответствии с тремя приведенными предложениями) и по три значения (соответственно 0 баллов, 5 баллов, 10 баллов и 5%, 15%, 25%) для центров функций принадлежности входов и выходов. Построим данную систему, используя алгоритм **Мамдани**, описывая требуемые действия по пунктам.

Задание к лабораторной работе 4

1. Командой **Fuzzy** запускаем FIS-редактор. По умолчанию предлагается алгоритм **Мамдани** (о чем говорит надпись в центральном белом блоке), здесь никаких изменений не требуется, но в системе должно быть два входа, поэтому через пункт меню **Edit> Add Variable Input** добавляем в системе этот второй вход (в окне редактора появляется второй желтый блок с именем **Input 2**). Делая далее однократный щелчок на блоке **Input 1**, меняем его имя на «сервис», завершая ввод нового имени нажатием клавиши. Аналогичным образом устанавливаем имя «еда» блоку **Input 2** и «чаевые»

- выходному блоку **Output 1**(справа вверху). Присвоим сразу же и имя всей системе, например TIP_ FIO**.** Зададим теперь функции принадлежности переменных. Программу – редактор функции принадлежности можно открыть тремя способами: выберем первый через пункт меню **View> Edit membership function**. Задание и редактирование функций принадлежности начнем с переменной «сервис». Сначала в полях **Range** и **Display Range** установим диапазон изменения и отображения этой переменной – от 0 до 10 (баллов), подтверждая ввод нажатием клавиши Enter. Затем через пункт меню Edit/Add MFs перейдем к диалоговому окну на рис. 2.

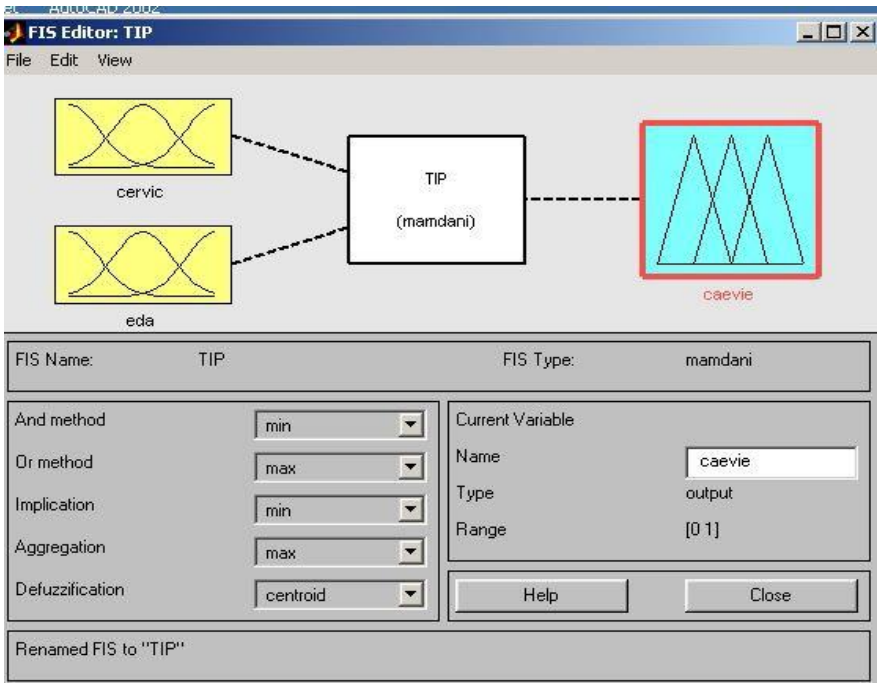


Рис. 2. Вид окна FIS Editor

Как показано на рис. 3, зададим три функции принадлежности гауссова типа (**gaussmf**). Нажмем кнопку OK и возвратимся в окно редактора функций принадлежности. Не изменяя размах и положение заданных функций, заменим только их имена на «плохой», «хороший» и «отличный». Щелчком на значке «еда» войдем в окно редактирования функций принадлежности для этой переменной. Зададим сначала диапазон ее изменения от 0 до 10, а затем, поступая как ранее, зададим две функции принадлежности трапецеидальной формы с параметрами соответственно [0 0 1 3] и [7 9 10 10] и именами «подгоревшая» и «превосходная».

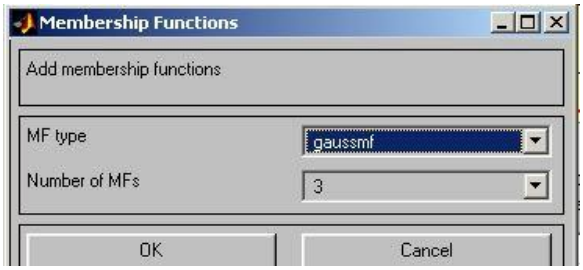


Рис. 3. Окно задания функции принадлежности пользователя

Для выходной переменной «чаевые» укажем сначала диапазон изменения (от 0 до 30), потом зададим три функции принадлежности треугольной формы с именами «малые», «средние» и « щедрые» так, как представлено на рис.4. Заметим, что можно, разумеется, задать и какие-либо другие функции или выбрать другие параметры.

Перейдем к конструированию правил. Для этого выберем пункт меню **View>Edit rules**. Далее ввод правил производится в соответствии с предложениями, описывающими задачу. Заметим, что в первом и третьем правилах в качестве «связки» в предпосылках правила необходимо использовать не «И» (and), а «ИЛИ» (or); при вводе второго правила, где отсутствует переменная «еда», для нее выбирается опция none. В результате формируется итоговый набор правил.

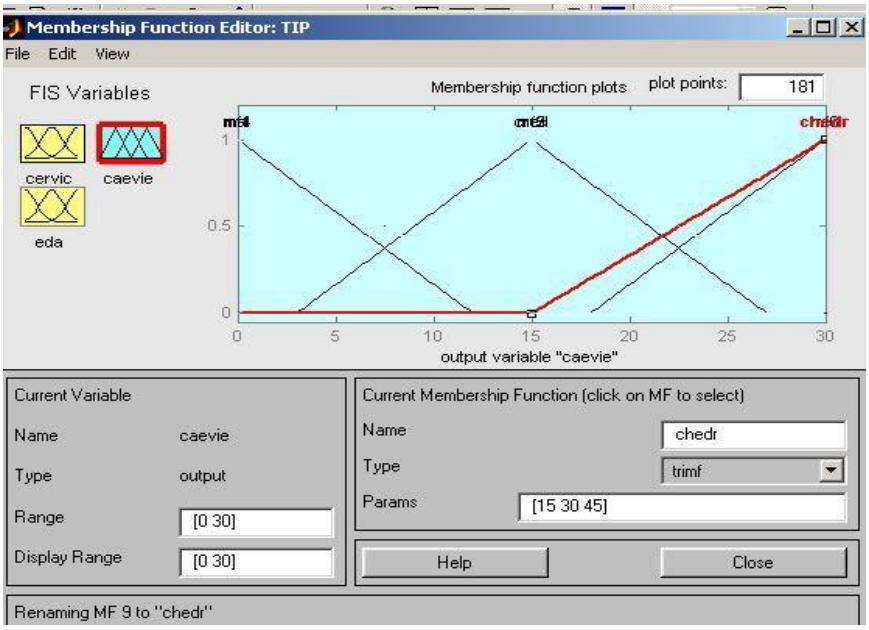


Рис. 4. Функции принадлежности переменной «чаевые»

Такая подробная (verbose) запись представляется достаточно понятной; единица в скобках после каждого правила указывает его «вес» (Weight), т.е. значимость правила. Данный вес можно менять, используя соответствующее поле в левой нижней части окна редактора правил. Правила представлены и в других формах – символической (symbolic) и индексной (indexed), при этом переход от одной формы к другой происходит с помощью меню Options> Format редактора правил. Вот как выглядят рассмотренные правила в символической форме:

1) (сервис==плохой)I(еда==подгоревшая)=>(чаевые==малые)(1);

2) (сервис==хороший)=>(чаевые==средние)(1);

3) (сервис==отличный)I(еда==превосходная)=>(чаевые==щедрые)(1). Правила понятны.

Наконец, самый сжатый формат представления правил – индексный – является тем форматом, который в действительности используется программой. В этом формате приведенные правила выглядят так:

o 1 1, 1 (1):2

o 2 0, 2 (1):2

o 3 2, 3 (1):2

Здесь первая колонка относится к первой входной переменной (соответственно первое, второе или третье возможное значение), вторая – ко второй, третья (после запятой) – к выходной переменной, цифра (после двоеточия) указывает тип «связки» (1 для «И», 2 для «ИЛИ»).

На этом соответственно конструирование экспертной системы закончено. Сохраним ее на диске под выбранным именем TIP.

Теперь самое время проверить систему в действии. Откроем (через пункт меню **View > View rules**) окно просмотра правил и установим значения переменных: сервис = 0 (то есть никуда не годный), еда = 10 (то есть превосходная). Увидим ответ: чаевые = 15 (то есть средние). Ну что ж, с системой не поспоришь, надо платить.

Можно проверить и другие варианты. В частности (может быть, не без удивления), выяснится, что нашей системой обслуживание ценится больше, чем качество еды: при наборе «сервис = 10, еда = 3» система советует определить размер чаевых в 23,9 %, в то время как набору «сервис = 3, еда = 10» размер чаевых по рекомендации системы — 16,6 % (от стоимости обеда). Впрочем, ничего удивительного здесь нет: это мы сами (не особенно подозревая об этом) заложили в систему соответствующие знания в виде совокупности приведенных правил.

Подтверждением отмеченной зависимости выходной переменной от входных может служить вид поверхности отклика, который представляется при выборе пункта меню View/View surface; обратите внимание, что с помощью мышки график можно поворачивать во все стороны.

В открывшемся окне, меняя имена переменных в полях ввода (X (input) и Y (input)), можно задать и просмотр одномерных зависимостей, например «чаевых» от «еды».

По приведенному выше примеру подготовим **свою экспертную систему**, протестируем

ее и сохраним под именем *ECFio*, оформив отчет в виде, представленном в начале лабораторной работы 4.

Лабораторная работа № 5. Часть 2. Создание пользовательских функций принадлежности

Цель работы. Работа с **Fuzzy Logic Toolbox** в режиме командной строки. Возможности работы в режиме командной строки. Функции систем нечеткого вывода. Функции сохранения, открытия и использования созданной системы.

Пакет **Fuzzy Logic Toolbox** располагает большим набором функций, исполняемых из командной строки MATLAB и позволяющих не использовать при работе с системами нечеткого вывода рассмотренные программы графического интерфейса. Все функции делятся на следующие группы:

1) вызов программ графического интерфейса;
2) задания функций принадлежности;
3) создание, редактирование, просмотр, открытие и сохранение систем нечеткого вывода;
4) дополнительные;
5) сервисные;
6) вызов диалоговых окон интерфейса;
7) блоки Simulink;
8) демонстрации возможностей пакета.

Функции вызова программ графического интерфейса

К этой группе относятся следующие функции:

• **fuzzy** — вызов FIS-редактора;

• **mfedit** — вызов редактора функций принадлежности;

• **ruleedit** — вызов редактора правил;

• **ruleview** — вызов программы просмотра правил;

• **surfview** — вызов программы просмотра поверхности отклика;

• **anfisedit** — вызов FIS-редактора (только для систем, использующих алгоритм Sugeno и имеющих одну выходную переменную);

• **findcluster** — вызов программы кластеризации.

Использование первых шести функций с аргументом (например, fuzzy(a), где **a** — имя переменной рабочего пространства, присвоенное системе нечеткого вывода) открывает соответствующую программу с одновременной загрузкой в нее рассматриваемой системы. Функция **findcluster** (имя_файла) открывает программу кластеризации с одновременной загрузкой указанного файла данных.

Чтение, использование и сохранение на диск созданной системы нечеткого вывода в режиме командной строки осуществляется следующими функциями:

readfis (‘имя_файла’)

evalfis (вектор_параметров, имя)

writefis (имя) или writefis (имя.'имя_файла')

Здесь имя_файла — наименование файла с написанной системой (без указания расширения), имя — идентификатор, который дан системе в рабочей среде MATLAB. Вектор_параметров — набор значений входов, для которых требуется рассчитать выход (возможна и матрица параметров, тогда результат расчетов — вектор в случае одной выходной переменной или матрица при нескольких таких переменных).

Задания к лабораторной работе

1. На примере ранее созданной и сохраненной на диске в виде файла с именем **Tip** экспертной системы для задачи о чаевых использовать системы нечеткого вывода с помощью графического интерфейса, приведенного выше:

```
>>a = readfis ('Tip');
```

>>out = evalfis ([1 2],a) out =

<результат>

>>writefis (a, ‘Tip’)

Выбрать следующие функции использования графического окна, которые позволяют использовать элементы графических изображений вне программ с графическим интерфейсом.

Команда (функция) **plotfis (a)** вызовет появление графического окна MATLAB с мнемоническим представлением разработанной системы (рис. 5.1).

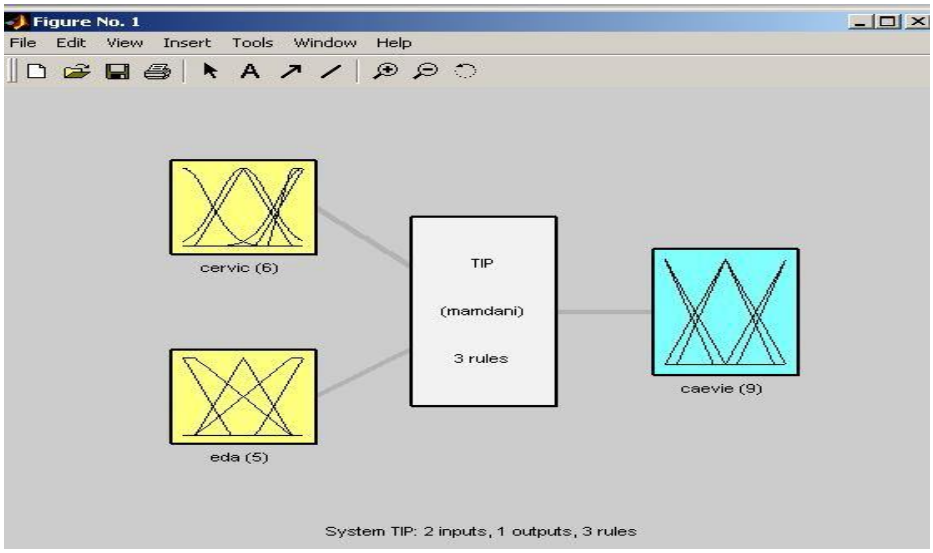


Рис 5.1. Мнемоническое представление системы нечеткого вывода Команда (функция) **gensurf (a)** делает то же самое, но применительно к поверхности отклика (рис. 5.2).

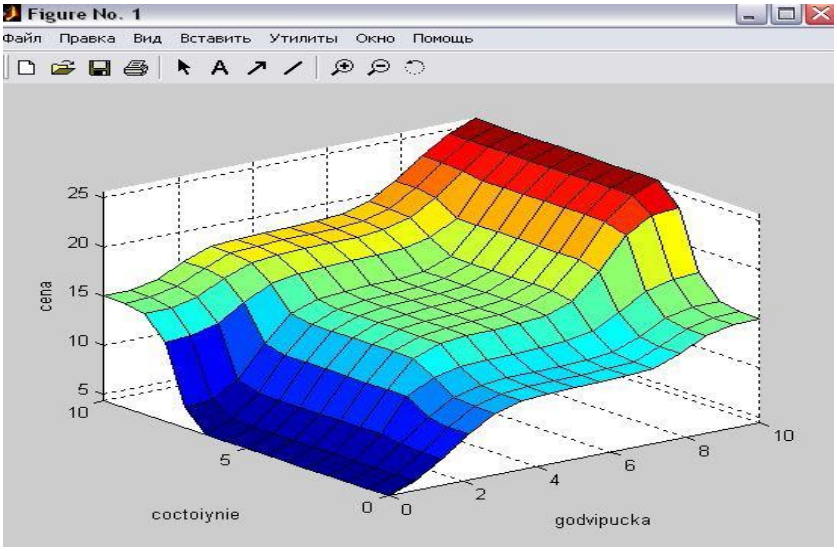
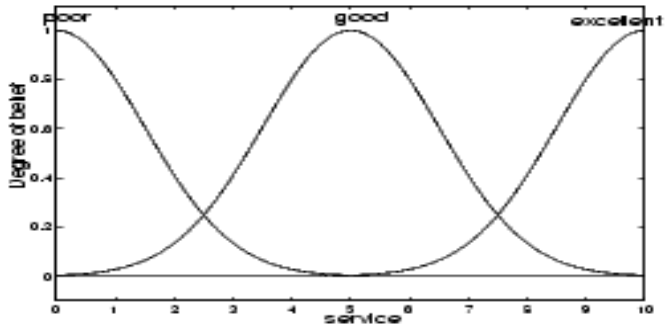


Рис.5.2. Результат выполнения функции gensurf (имя)

2. Далее выполнить пример построения экспертной системы для задачи о чаевых, используя функции создания, просмотра структуры и редактирования систем нечеткого вывода: функции **newfis** (новая система), **addvar** (добавить переменную), **addmf** (добавить функцию принадлежности) и **addrule** (добавить правило).

3. Сконструировать систему нечеткого вывода целиком в режиме командной строки MATLAB.

```
a = newfis('tipper');
a = addvar(a,'input','service',[0 10]);
a = addmf(a,'input',1,'poor','gaussmf',[1.5 0]);
a = addmf(a,'input',1,'good','gaussmf',[1.5 5]);
a = addmf(a,'input',1,'excellent','gaussmf',[1.5 10]);
plotmf(a,'input',1)
```



Затем точно так же указать все для «еды» и для «чаевых», только заменить ‘input’ на ‘output’.

ruleList=[1 1 1 1 1
1 2 2 1 1];
a = addrule(a,ruleList).

Далее при необходимости просмотра элементов структуры (системы *c* идентификатором *a*) в режиме командной строки следует ввести команду вида *a*. Параметр, например

>> a.type Получим ответ: ans = mamdani.

Получение информации по всем элементам структуры обеспечивается функцией **getfis** (a).

Функция **setfis** (a) в определенном смысле противоположна функции **getfis** (a) и позволяет изменять параметры.

Полный просмотр структуры системы нечеткого вывода осуществляется функцией **showfis** (a).

Просмотр правил, включенных в систему нечеткого вывода, осуществляется с помощью функции **showrule** (a).

a = readfis ('tipper'); showrule (a,1)
ans =
1. If (service is poor) or (food is rancid) then (tip is cheap) (1) showrule (a,2)
ans =
2. If (service is good) then (tip is average) (1) Showrule (a,[3 1],'symbolic')
ans =
3. (service==excellent) | (food==delicious) => (tip=generous) (1)
1. (service==poor) | (food==rancid) => (tip=cheap) (1) Showrule (a,1:3,'indexed')
ans =
1 1, 1 (1) : 2
2 0, 2 (1) : 1
3 2, 3 (1) : 2.

4. Повторить весь процесс с применением указанных программ для своей экспертной системы, созданной в лабораторной работе 4, на примере построения системы в задаче о чаевых.

Лабораторная работа № 6. Часть 1. Нечеткие множества. операции над нечеткими множествами

Нечеткие множества и нечеткая логика

Нечетким множеством A называется совокупность пар $A = \{\langle x, \mu_A(x)\rangle | x \in U\}$, где μ_A — функция принадлежности, т.е. $\mu_A : U \to [0,1]$, характеристическая функция множества $A \subseteq U$, значения которой указывают, является ли $x \in U$ элементом множества A, U - так называемое универсальное множество, из элементов которого образованы все остальные множества, рассматриваемые в данном классе задач.

Значение $\mu_A(x)$ называется степенью принадлежности элемента x нечеткому множеству A.

Операции над нечеткими множествами

Аналогично четким множествам над нечеткими множествами можно производить ряд операций, которые могут определяться 3 способами (таблица 12).

Таблица 12. Виды определений операций над нечеткими множествами.

Максиминные	$\mu_{A\cup B}(x) = \max\{\mu_A(x), \mu_B(x)\},$ $\mu_{A\cap B}(x) = \min\{\mu_A(x), \mu_B(x)\}.$
Алгебраические	$\mu_{A\cup B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\cdot\mu_B(x),$ $\mu_{A\cap B}(x) = \mu_A(x)\cdot\mu_B(x).$
Ограниченные	$\mu_{A\cup B}(x) = \min\{1, \mu_A(x) + \mu_B(x)\},$ $\mu_{A\cap B}(x) = \max\{0, \mu_A(x) + \mu_B(x) - 1\}.$

Дополнение нечеткого множества во всех трех случаях определяется одинаково:

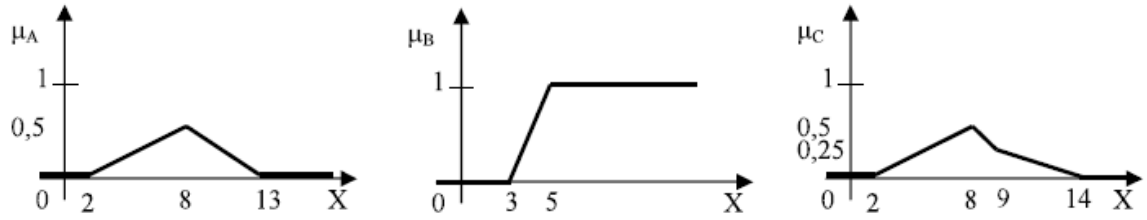
$$\mu_{\overline{A}}(x) = 1 - \mu_A(x).$$

При графическом определении функций принадлежности объединенного множества необходимо в каждой точке множества выбрать максимальное значение из двух (точку того графика, который выше) и объединить все полученные точки в график, который и будет отображением новой функции принадлежности. Пересечение аналогично объединению, только выбирается минимальное значение в каждой точке. При построении дополнения необходимо зеркально отобразить график от оси, параллельной оси абсцисс и проходящей через точку 0,5 оси ординат.

Пример решения задачи

Задача. Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого

множества D = $\overline{A} \cap (A \cup C \cup B)$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений.

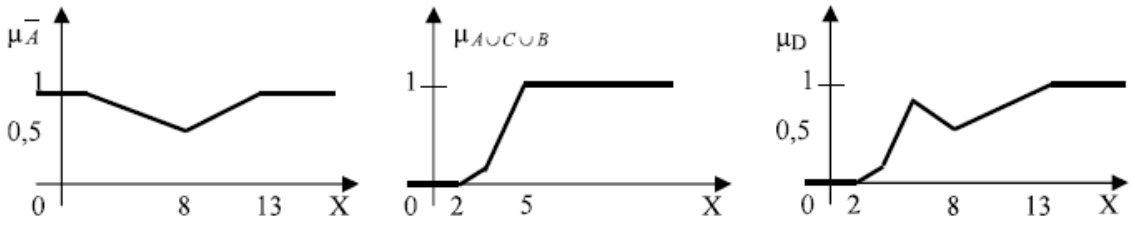


Описание процесса решения. Для построения функции принадлежности нового множества необходимо:

1) Определить последовательность выполнения операций в формуле.
2) Построить на отдельных графиках промежуточные множества, согласно определенной последовательности действий. Свести промежуточные множества на одном графике и определить итоговую функцию принадлежности.
3) Используя определенный в задаче метод, определить аналитически степень принадлежности элемента, входящего в ядро итогового множества.
4) Проверить аналитические вычисления по построенному графику функции принадлежности.

Решение.

1) Множество D = $\overline{A} \cap (A \cup C \cup B)$, значит, последовательность операций будет следующей: $\overline{A}$, $A \cup C \cup B$, $\overline{A} \cap (A \cup C \cup B)$.
2) Построим согласно этой последовательности операций графики функций принадлежности:



3) Ядро множества D состоит из элементов из интервала (2,13). Выберем элемент 8.

$$\mu_A(8) = 0.5;$$

$$\mu_B(8) = 1;$$

$$\mu_C(8) = 0.5;$$

$$\mu_{\overline{A}}(8) = 1 - \mu_A(8) = 1 - 0.5 = 0.5;$$

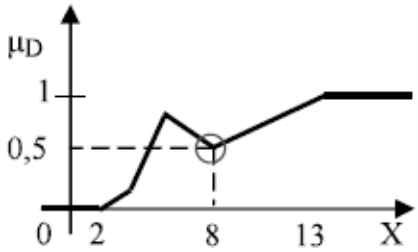
$$\mu_{\overline{C}}(8) = 1 - \mu_C(8) = 1 - 0.5 = 0.5;$$

$$\mu_{A \cup C}(8) = \min\{1, \mu_C(8) + \mu_A(8)\} = \min\{1, 0.5 + 0.5)\} = 1;$$

$$\mu_{A \cup C \cup B}(8) = \min\{1, \mu_{A \cup C}(8) + \mu_B(8)\} = \min\{1, 1 + 1)\} = 1;$$

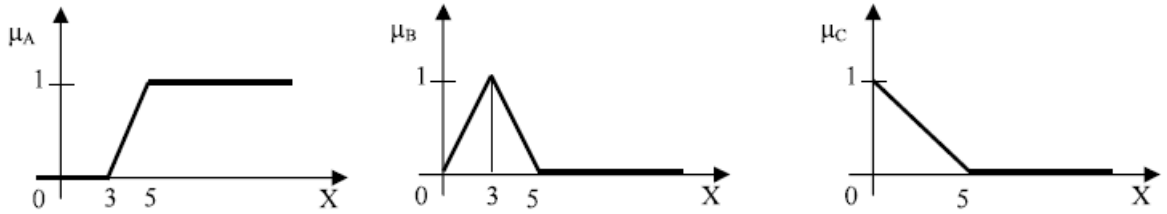
$$\mu_{\overline{A} \cap (A \cup C \cup B)}(8) = \max\{0, \mu_{\overline{A}}(8) + \mu_{A \cup C \cup B}(8) - 1\} = \max\{0, 0.5 + 1 - 1\} = 0.5.$$

4) $\mu_D(8) = 0.5;$

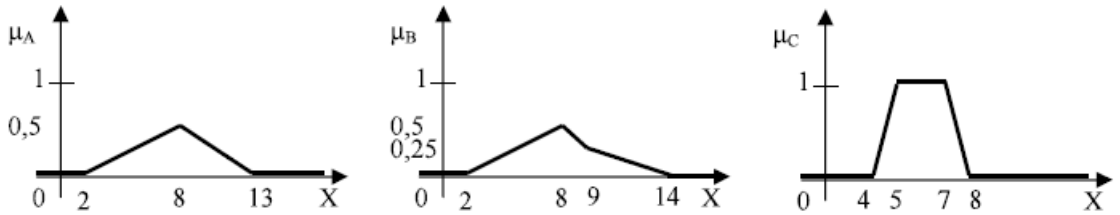


Задачи

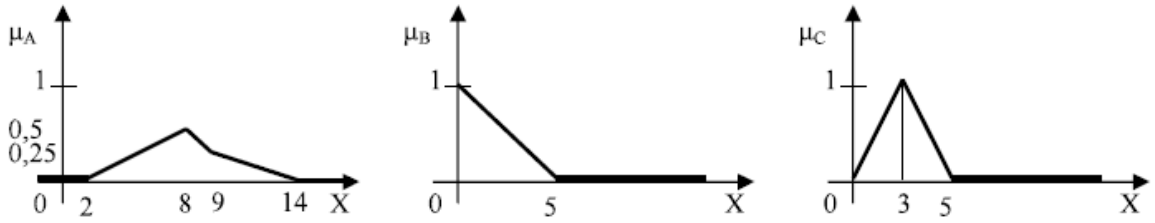
1) Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества D = $A \cup B \cap C$ и определить степень принадлежности одного элемента множеству D, используя максиминный способ.



2) Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества D = $A \cup B \cap C$ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ.

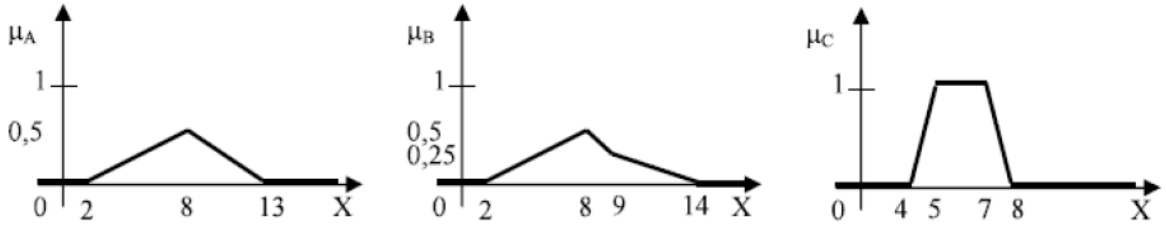


3) Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества D = $A \cup B \cap C$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений.

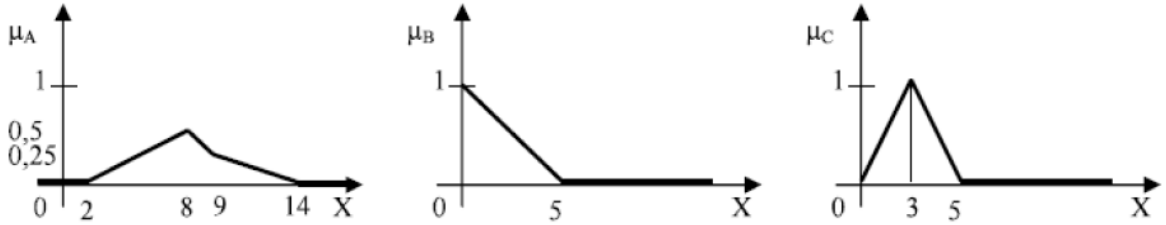


4) Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества D = $A \cup \overline{B} \cap C$ и определить степень принадлежности одного элемента множеству D, используя максиминный способ.

5) Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества D = $A \cup \overline{B} \cap C$ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ.



6) Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества D = $A \cup \overline{B} \cap C$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений.



Цель работы: освоить проектирование нечетких систем в пакете *Fuzzy Logic Toolbox* и *Symbolic Math Toolbox* вычислительной среды *MATLAB*.

Порядок выполнения работы

1. Получить задание по одному из вариантов таблицы (п. 4).
2. Подготовить нечеткие правила в соответствии с графическим представлением функций задания.
3. Трехмерное изображение выбранной зависимости построить, воспользовавшись программными средствами *MATLAB*.

При этом можно руководствоваться программой нижеприведенного примера, которая моделирует зависимость $y=x^2 \cdot sin(x-1)$.

% Построение графика функции *y*=*x*1^2sin*(*x*2-1)**

% в области *x1* =[-6, 4], *x2*=[-4.4, 1.7].

n*=15; % количество точек дискретизации *x1*=*linspace*(-6, 4, n); *x2*=*linspace*(-4.4, 1.7, *n*); *y*=*zeros*(*n*, *n*);*for j*=1:*n

***y*(*j*,:)=*x*1.^2**sin*(*x*2(*j*)-1); *end surf*(*x*1, *x*2, *y*)**

***xlabel*('*x*_1'); *ylabel*('*x*_2'); *zlabel*('*y*'); *title*('Искомая зависимость')**

Сформируем нечеткие правила, соответствующие зависимости выходной переменной от входных по полученной поверхности рис. 1.

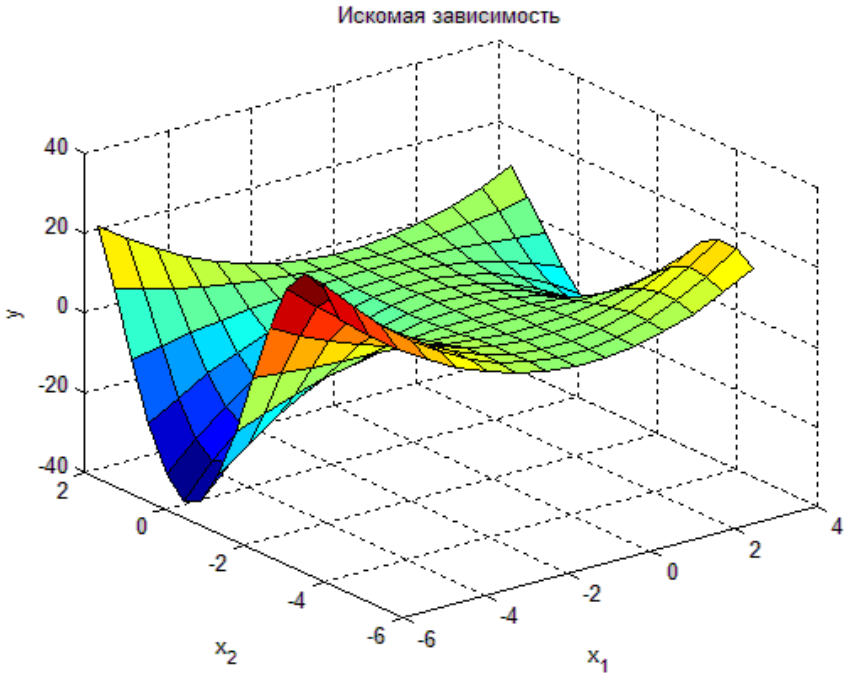


Рис. 1. Графическое представление функции $y=x_1^2 \cdot sin(x_2-1)$

Можно воспользоваться и возможностями пакета *Symbolic Math Toolbox*, введя, например, следующие команды:

***syms x y* % создаем символьные переменные**

***z*=*x*^2**sin*(*y*-1); *ezsurf*(*z*,[-2,2])**

Для лингвистической оценки входных переменных x1 и x2 выберем по три терма - низкий, средний, высокий; для выходной y пять термов *низкий, ниже среднего, средний, выше среднего, высокий*.

Получим, например, следующие правила:

1) ЕСЛИ x_1 = *низкий* И x_2 = *низкий*, ТО y = *высокий*;

2) ЕСЛИ x_1 = *низкий* И x_2 = *средний*, ТО y = *низкий*;

3) ЕСЛИ x_1 = *низкий* И x_2 = *высокий*, ТО y = *высокий*;

4) ЕСЛИ x_1 = *средний,* ТО y = *средний*;

5) ЕСЛИ x_1 = *высокий* И x_2 = *низкий*, ТО y = *выше среднего*;

6) ЕСЛИ x_1 = *высокий* И x_2 = *средний*, ТО y = *ниже среднего*;

7) ЕСЛИ x_1 = *высокий* И x_2 = *высокий*, ТО y = *выше среднего*.

4. Спроектировать нечеткую систему, выполнив следующую последовательность шагов.

1. Открыть *FIS*–редактор, напечатав слово *fuzzy* в командной строке.

2. В появившемся графическом окне *FIS Editor* добавим вторую входную переменную. Для этого в меню *Edit* выбираем команду *Add input.*

3. Переименуем первую входную переменную. Для этого сделаем щелчок левой кнопкой мыши на блоке *Input 1*, введем новое обозначение x_1 в поле редактирования имени текущей переменной и нажмем <*Enter*>.

4. Переименуем вторую входную переменную, введя x_2 на блоке *Input 2*.

5. Переименуем выходную переменную. Для этого щелкнем мышкой на блоке *Output 1*. Введем новое обозначение y в поле редактирования имени текущей переменной; нажмем <*Enter*>.

6. Зададим имя системы. Для этого в меню *File* выберем в подменю *Export* команду *To File* и введем имя файла, например, *Lab_2*.

7. Перейдем в редактор функций принадлежности. Для этого сделаем двойной щелчок левой кнопкой мыши на блоке x_1 и зададим диапазон изменения переменной x_1, напечатав -6 4 в поле *Range* (рис. 2).

8. Зададим функции принадлежности переменной x_1. Для лингвистической оценки этой переменной будем использовать три терма с треугольными функциями принадлежности. Эти функции установлены по умолчанию, поэтому переходим к следующему шагу.

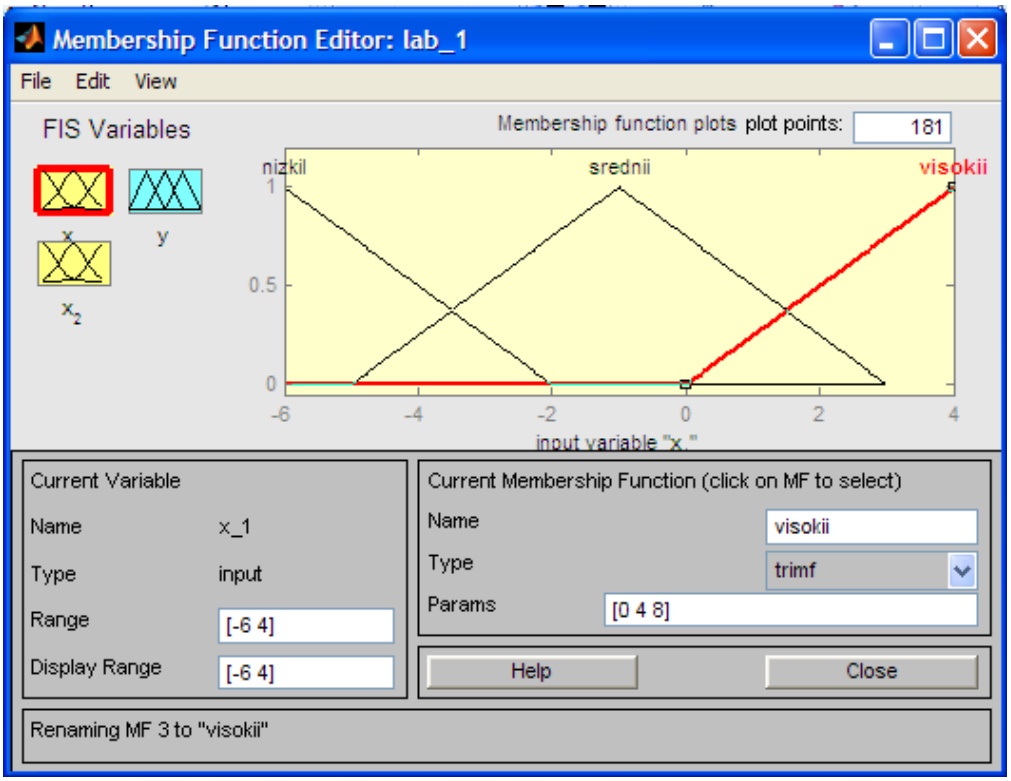


Рис. 2. Функции принадлежности переменной *x1*

9. Зададим наименования термов переменной ***x1***. Для этого щелкнем мышкой по графику первой функции принадлежности (см. рис. 2). График активной функции принадлежности выделяется красной жирной линией. Затем введем наименование терма Низкий в поле *Name* и нажмем <*Enter*>. Щелкнем мышкой по графику второй функции принадлежности, введем наименование терма Средний в поле *Name* и нажмем <*Enter*>. Щелкнем мышкой по графику третьей функции принадлежности, введем наименование терма Высокий в поле *Name* и нажмем <*Enter*>.

10. Зададим функции принадлежности переменной x_2. Для этого активизируем переменную x_2 щелчком мышкой по блоку x_2. Зададим диапазон изменения переменной ***x2***. Для этого напечатаем -4.4 1.7 в поле *Range* и нажмем <*Enter*>. Для лингвистической оценки этой переменной будем использовать, как и ранее, три терма с треугольными функциями принадлежности. Они установлены по умолчанию, поэтому переходим к следующему шагу.

11. По аналогии с шагом 9 зададим следующие наименования термов переменной x_2: Низкий, Средний, Высокий.

12. Зададим функции принадлежности переменной *y*. Для лингвистической оценки этой переменной будем использовать пять термов с гауссовыми функциями принадлежности. Для этого щелчком мыши по блоку *y* активизируем переменную *y*. Зададим диапазон изменения переменной *y*. Для этого напечатаем -50 50 в поле *Range* (рис. 1.3) и нажмем <*Enter*>. Затем в меню *Edit* выберем команду *Remove All MFs* для удаления установленных по умолчанию функций принадлежности. После этого в меню *Edit* выберем команду *Add MF* В появившемся диалоговом окне выберем тип функции принадлежности *gaussmf* в поле *MF type* и пять термов в поле *Number MFs*. После ввода функций принадлежности редактор активизирует первую входную переменную, поэтому для продолжения работы щелкнем мышкой по пиктограмме ***y***.

13. По аналогии с шагом 9 зададим следующие наименования термов переменной *y*: *Низкий, Ниже среднего, Средний, Выше среднего, Высокий*. В результате получим графическое окно, изображенное на рис. 1.3.

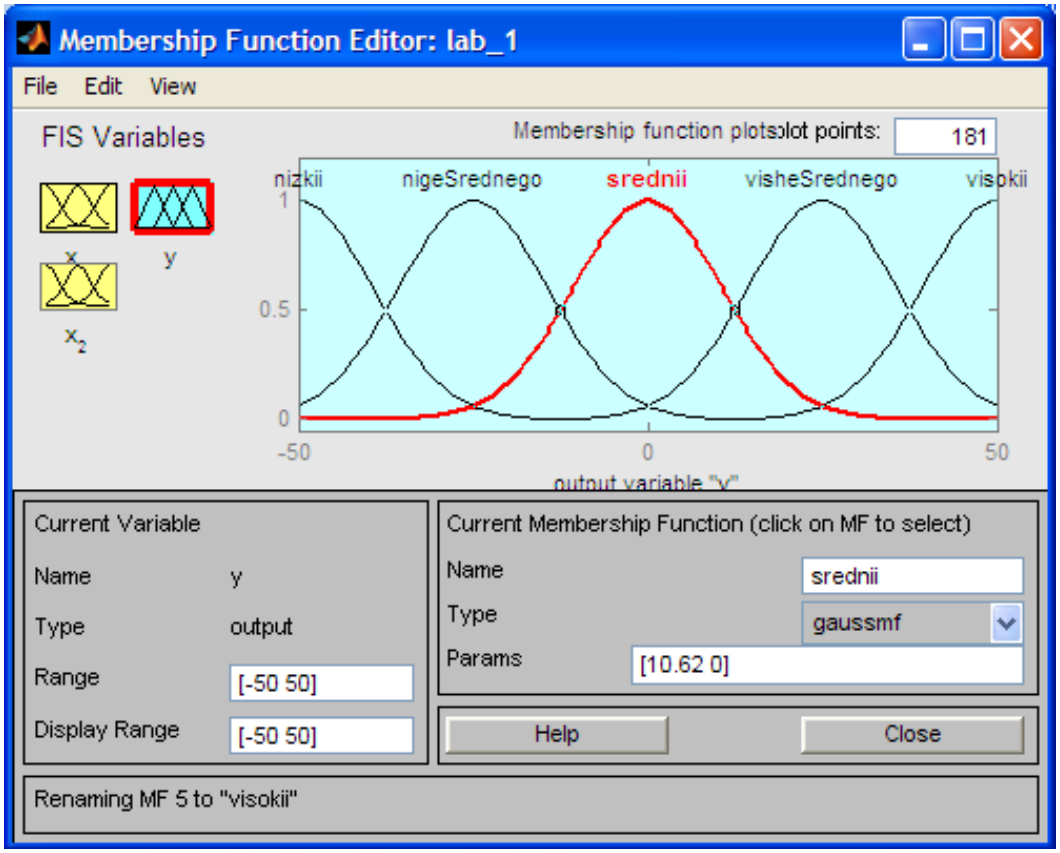


Рис.3. Функции принадлежности переменной *y*

14. Перейдем в редактор базы знаний *Rule Editor*. Для этого в меню *Edit* выберем команду *Rules...* Для ввода правила выбираем в меню соответствующую комбинацию термов и нажимаем кнопку *Add rule*. На рис. 1.4 изображено окно редактора базы знаний после ввода всех семи правил. В конце правил в скобках указаны весовые коэффициенты.

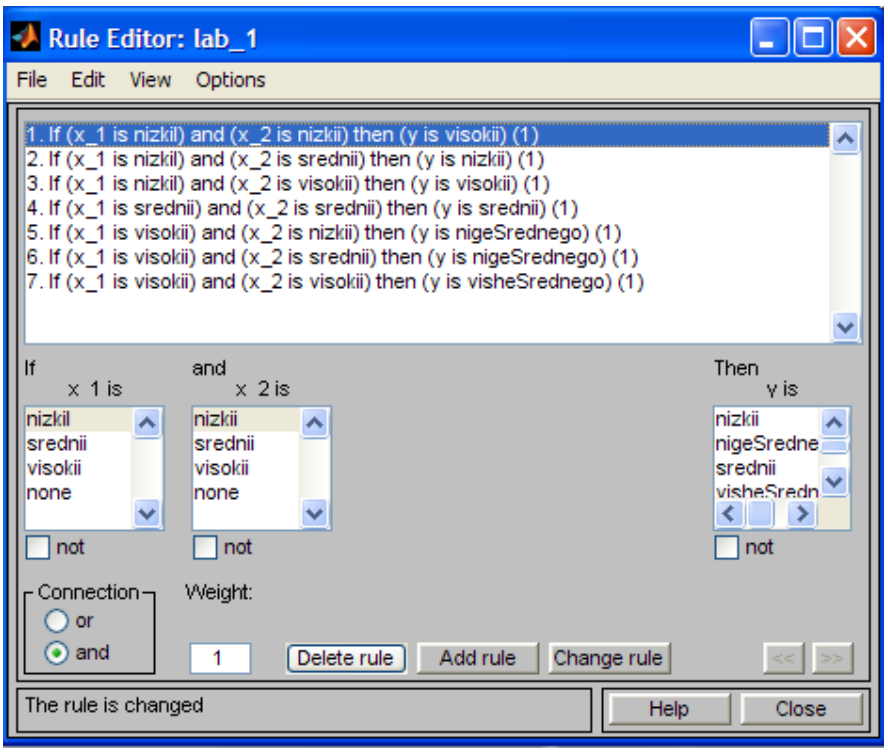


Рис. 4. Нечеткая база знаний *Mamdani*

На рис. 5 приведено окно визуализации нечеткого вывода. Окно активизируется командой *Rules* меню *View*. В поле *Input* указываются значения входных переменных, для которых выполняется нечеткий логический вывод.

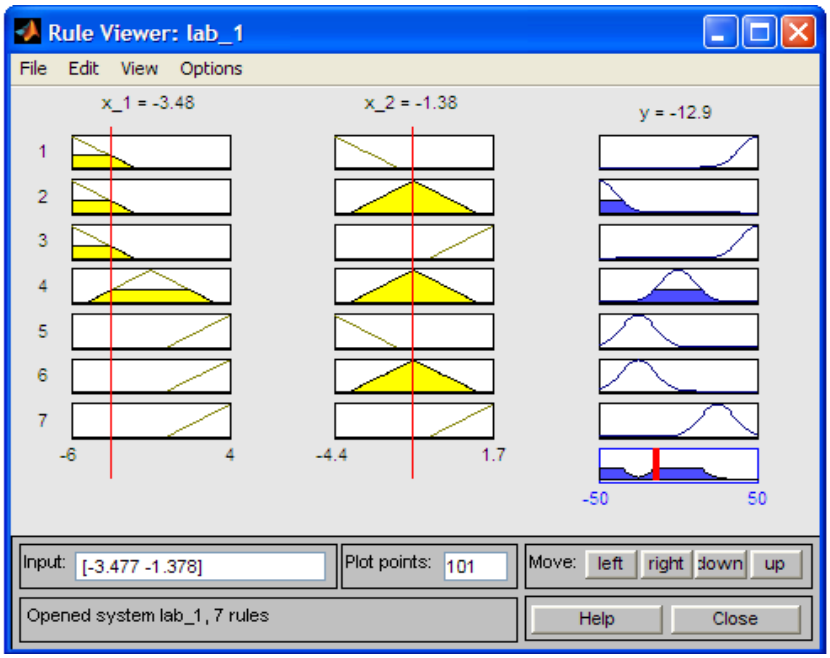


Рис. 5. Визуализация нечеткого вывода *Mamdani*

На рис. 6 приведена поверхность «входы выход», соответствующая синтезированной нечеткой системе. Окно выводится по команде *Surface* меню *View*.

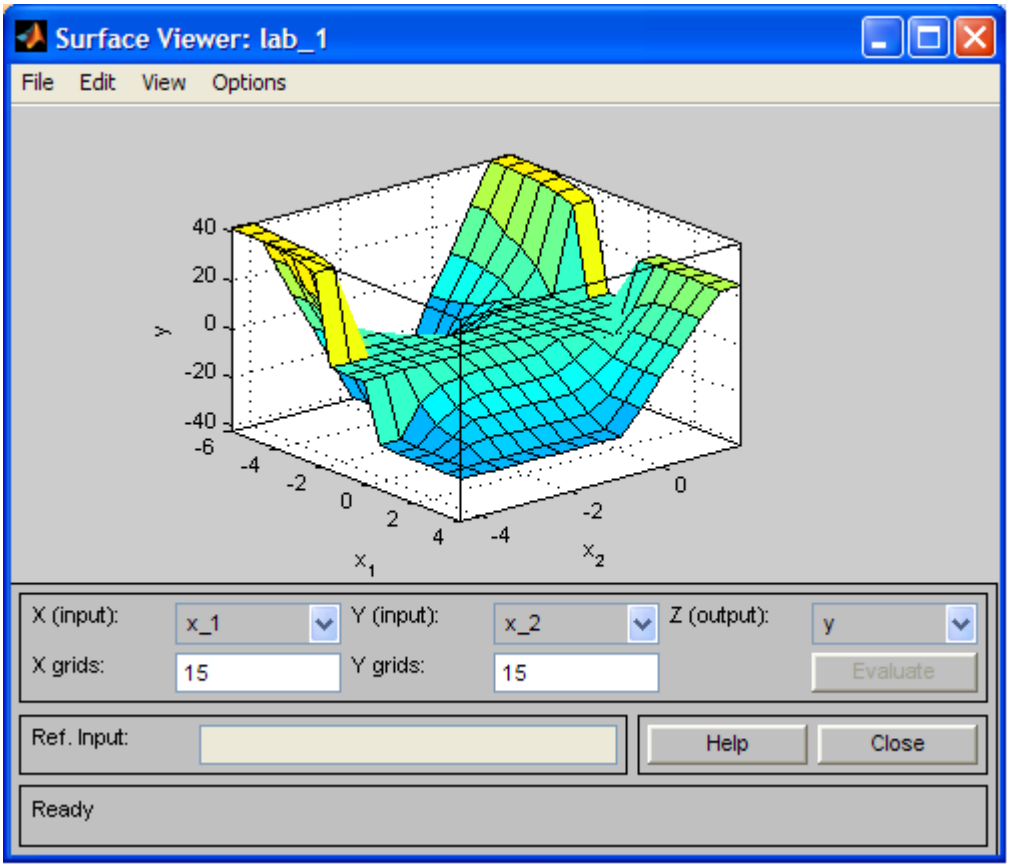


Рис. 6. Поверхность «входы выход» для базы знаний в *Surface Viewer*

15. Сохраним созданную систему. Для этого в меню *File* выберем в подменю *Export* команду *To File.*

При сравнении поверхностей на рис. 1 и 6 можно сделать выводы относительно качества описания нечеткими правилами моделируемой нелинейной зависимости.

Варианты заданий

Таблица 1. №	Вид зависимости	Диапазон изм.
1	Z = x.^2-y.^2	x,y [-1 1]
2	Z = x.^3+y.^2	x,y [-1 1]
3	Z = exp(-x.^2-y.^2)	x,y [-1 1]
4	Z = exp(-x.^2+y.^2)	x,y [-2 2]
5	Z = x*y*sin(x^2+y^2)	x,y [-2 2]
6	Z = x.^2*sin(y-1)	x,y [-2 2]
7	Z = y.^2*sin(x)	x,y [-2 2]
8	Z = y.^2*cos(x)	x,y [-2 2]
9	Z = y.^2*cos(x)^2	x,y [-1, 1]
10	Z = 4*cos(x)/y	x,y [0.5 3.14]
11	Z = (x-y)/(x+y)	x,y [1 10]
12	Z = exp(-x.^2)+exp(-y.^2)	x,y [-1 1]

Содержание отчета

1. Исходная функция варианта задания и ее графическое представление.

2. Лингвистические правила решений.
3. Описание последовательности действий при проектировании нечеткой системы.
4. Анализ результатов нечеткого вывода.
5. Оценка качества описания нечеткими правилами моделируемой нелинейной зависимости.

Лабораторная работа № 7. Построение нечеткой аппроксимирующей системы в пакете Fuzzy Logic Toolbox

Цель работы: изучение основных функций пакета *Fuzzy Logic Toolbox* программной среды *MATLAB*, а также приобретение навыков построения нечеткой аппроксимирующей системы.

Порядок выполнения работы

Командой (функцией) *Fuzzy* из режима командной строки запускается основная интерфейсная программа пакета *Fuzzy Logic* – редактор нечеткой системы вывода (*Fuzzy Inference System Editor, FIS Editor, FIS*-редактор).

Вид открывающегося при этом окна приведен на рис.1. Главное меню редактора содержит позиции:

File – работа с файлами моделей (их создание, сохранение, считывание и печать);

Edit – операции редактирования (добавление и исключение входных и выходных переменных);

View – переход к дополнительному инструментарию.

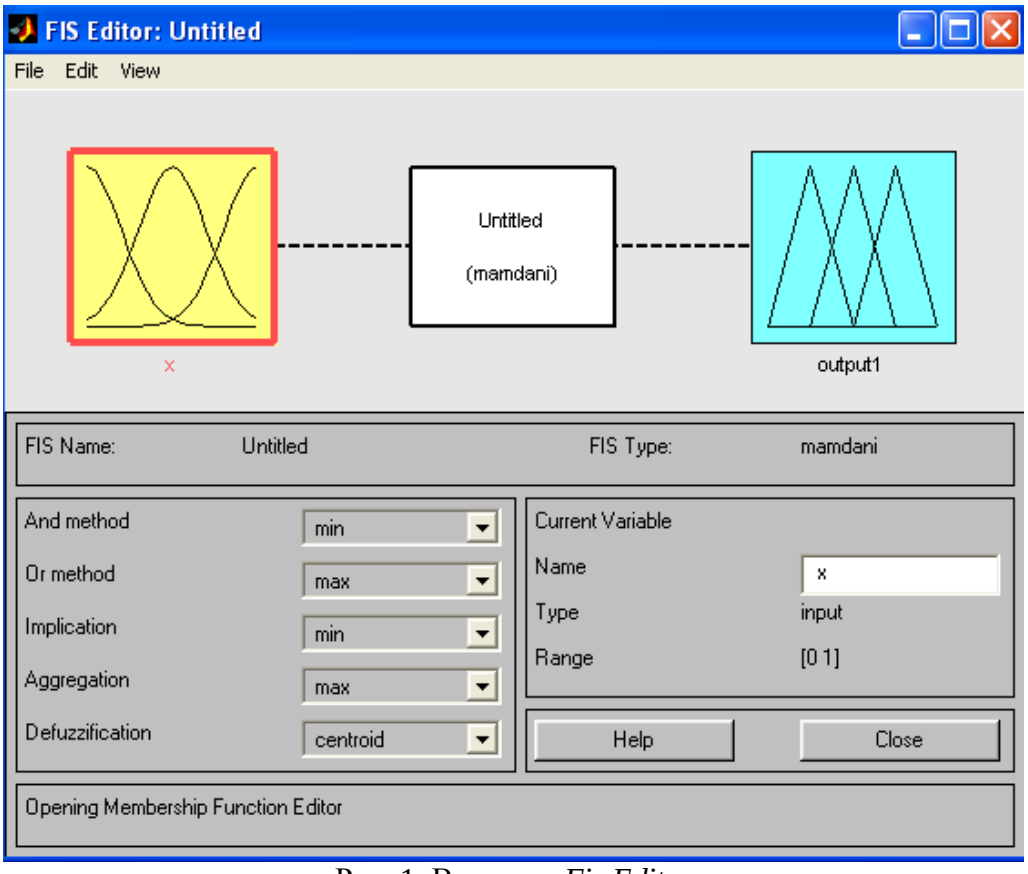


Рис. 1. Вид окна *Fis Editor*

Попробуем сконструировать нечеткую систему, отображающую зависимость между переменными x и y, заданную с помощью табл. 1 (легко видеть, что представленные в таблице данные отражают зависимость $y = x^2$).

Таблица 1. Значения x и y

x	-1	0.6	0	0.4	1
y	1	0.36	0	0.16	1

Требуемые действия отобразим следующими пунктами.

1. В позиции меню *File* выбираем опцию *New Sugeno FIS* (новая система типа *Sugeno*), при этом в блоке, отображаемом белым квадратом, в верхней части окна редактора появится надпись *Untitled2* (*Sugeno*).

2. Щелкнем левой кнопкой мыши по блоку, озаглавленному *input* 1 (вход 1). Затем в правой части редактора в поле, озаглавленном *Name* (Имя), вместо *input* 1 введем обозначение нашего аргумента, т.е. *x*. Обратим внимание, что если теперь сделать где-нибудь (вне блоков редактора) однократный щелчок мыши, то имя отмеченного блока изменится на *x*; то же достигается нажатием после ввода клавиши *Enter*.

3. Дважды щелкнем по этому блоку. Перед нами откроется окно редактора функций принадлежности – *Membership Function Editor*

(см. рис. 2.2). Войдем в позицию меню *Edit* данного редактора и выберем в нем опцию *Add MFs* (*Add Membership Funcions* – Добавить функций принадлежности). При этом появится диалоговое окно (рис. 2.3), позволяющее задать тип (*MF type*) и количество (*Number of MFs*) функций принадлежности (в данном случае все относится к входному сигналу, т. е. к переменной *x)*. Выберем гауссовы функции принадлежности (*gaussmf*), а их количество зададим равным пяти – по числу значений аргумента в табл. 1. Подтвердим ввод информации нажатием кнопки ОК, после чего произойдет возврат к окну редактора функций принадлежности.

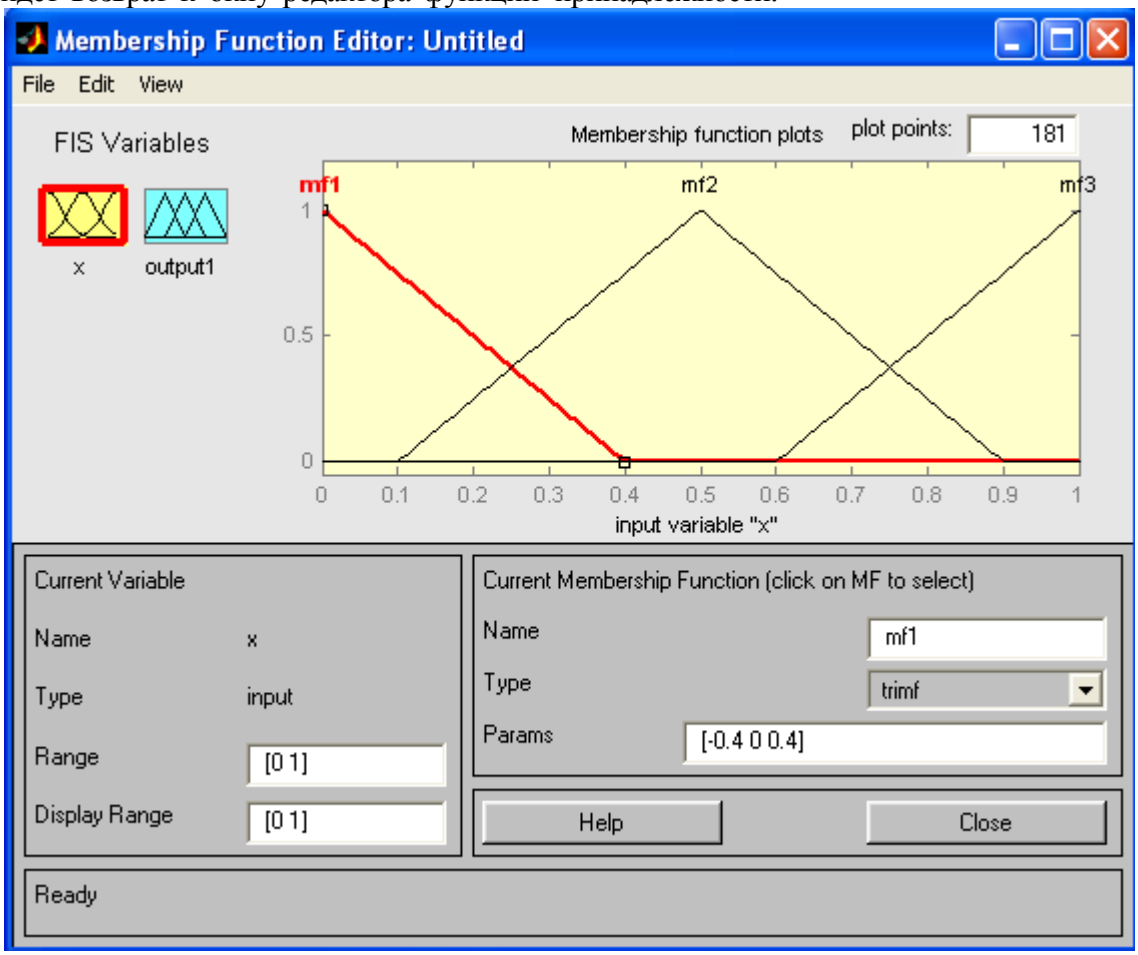


Рис. 2. Окно редактора функций принадлежности

4. В поле *Range* (Диапазон) установим диапазон изменения *x* от -1 до +1, т.е. диапазон, соответствующий табл. 1. Щелкнем затем левой кнопкой мыши где-нибудь в поле редактора (или нажмем клавишу ввода Enter). Обратим внимание, что после этого произойдет соответствующее изменение диапазона в поле *Display Range* (Диапазон дисплея).

5. Обратимся к графикам заданных нами функций принадлежности, изображенным в верхней части окна редактора функций принадлежности. Заметим, что для успешного решения поставленной

задачи необходимо чтобы ординаты максимумов этих функций совпадали с заданными значениями аргумента *x*. Для левой, центральной и правой функций такое условие выполнено, но две другие необходимо «подвинуть» вдоль оси абсцисс. «Передвижка» делается весьма просто: подводим курсор к нужной кривой и щелкаем левой кнопкой мыши. Кривая выбирается, окрашиваясь в красный цвет, после чего с помощью курсора ее и можно подвинуть в нужную сторону (более точную установку можно провести, изменяя числовые значения в поле *Params* (Параметры) – в данном случае каждой функции принадлежности соответствуют два параметра, при этом первый определяет размах кривой, а второй – положение ее центра). Для выбранной кривой, кроме этого, в поле Name можно изменять имя (завершая ввод каждого имени нажатием клавиши Enter).

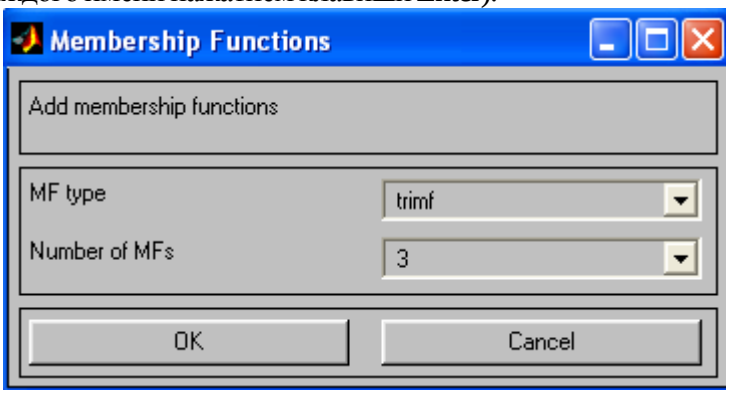


Рис.3. Диалоговое окно задания типа и количества функций принадлежности

Проделаем требуемые перемещения кривых и зададим всем пяти кривым новые имена, например:

- самой левой – *bn*,
- следующей – *n*,
- центральной – *z*,
- следующей за ней справа – *p*,
- самой левой – *bp*.

Нажмем кнопку *Close* и выйдем из редактора функций принадлежности, возвратившись при этом в окно редактора нечеткой системы (*FIS Editor*).

6. Сделаем однократный щелчок левой кнопкой мыши по голубому квадрату (блоку), озаглавленному *output* 1 (выход 1). В окошке *Name* заменим имя *output* 1 на *y* (как в пункте 2).

7. Дважды щелкнем по отмеченному блоку и перейдем к программе – редактору функций принадлежности. В позиции меню *Edit* выберем опцию *Add MFs*. Появляющееся диалоговое окно вида рис. 3 позволяет задать теперь в качестве функций принадлежности только линейные (*linear*) или постоянные (*constant*) – в зависимости от того, какой алгоритм

Sugeno (1-го или 0-го порядка) мы выбираем. Если в вашем компьютере установлена версия, в которой нет данных функций принадлежностей, то можно оставить по умолчанию – *trimf*. Это, конечно, повлияет на результат, поэтому можно поэкспериментировать, изменяя тип функций принадлежности.

В рассматриваемой задаче необходимо выбрать постоянные функции принадлежности с общим числом 4 (по числу различных значений *y* в табл. 2.1). Подтвердим введенные данные нажатием кнопки ОК, после чего произойдет возврат в окно редактора функций принадлежнсти.

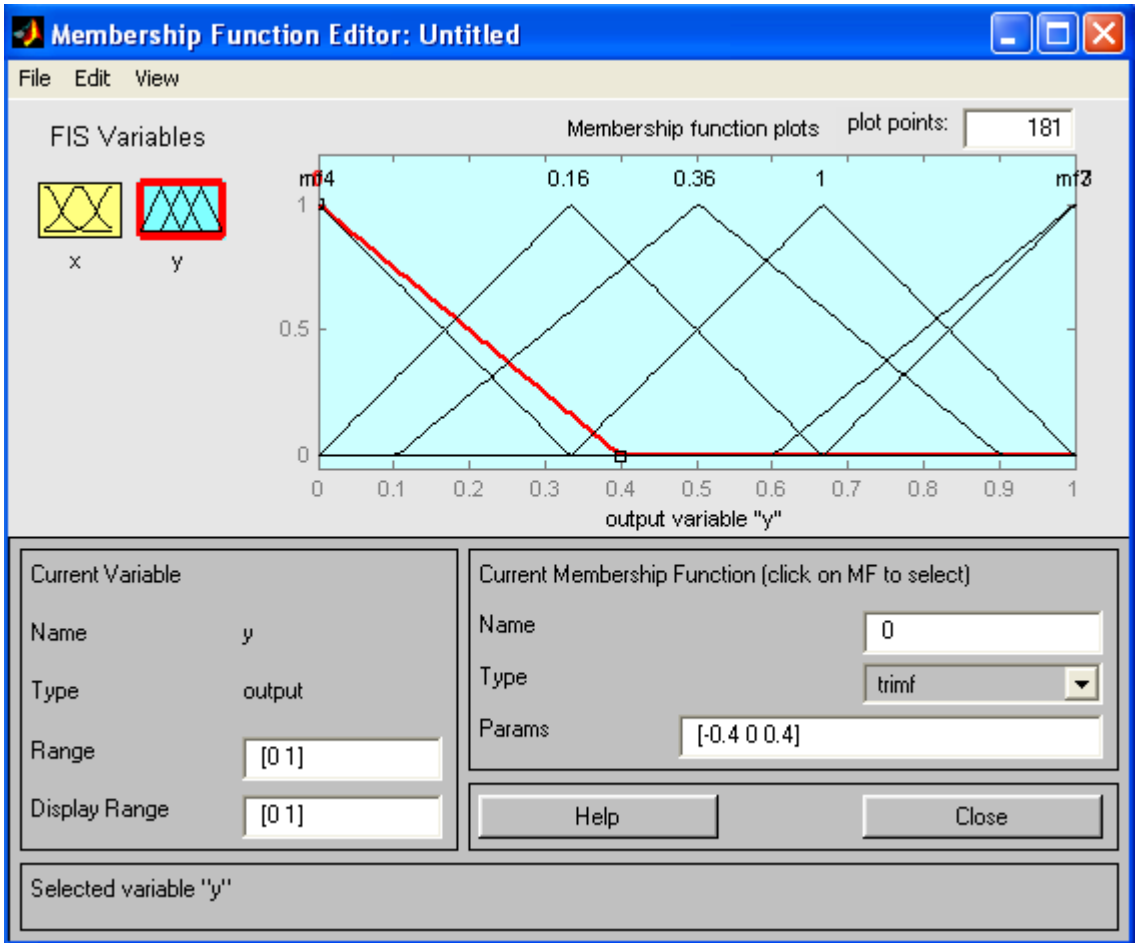


Рис. 2.4. Параметры функций принадлежности переменной *y*

8. Обратим внимание, что здесь диапазон (*Range*) изменения, устанавливаемый по умолчанию – [0, 1], менять не нужно. Изменим лишь имена функций принадлежности (их графики при использовании алгоритма *Sugeno* для выходных переменных не приводятся), например, задав их как соответствующие числовые значения *y,* т.е. 0, 0.16, 0.36, 1; одновременно эти же числовые значение введем в поле *Params* (рис. 2.4). Затем закроем окно нажатием кнопки *Close* и вернемся в окно *FIS*-редактора.

9. Дважды щелкнем левой кнопкой мыши по среднему (белому) блоку, при этом раскроется окно еще одной программы – редактора правил (*Rule Editor*). Введем соответствующие правила. При вводе каждого правила необходимо обозначить соответствие между каждой функцией принадлежности аргумента *x* и числовым значением *y*. Кривая, обозначенная нами *bn*, соответствует $x = -1$, т.е. $y = 1$. Выберем, поэтому в левом поле (с заголовком *x is bn)*, а в правом 1 и нажмем кнопку *Add rule* (Добавить правило). Введенное правило появится в окне правил и будет представлять собой запись:

If (*x* is *bn*) *then* (*y* is 1)

Аналогично поступим для всех других значений *x*, в результате чего сформируется набор из 5 правил (см. рис. 2.5). Закроем окно редактора правил и возвратимся в окно *FIS*-редактора. Построение системы закончено и можно начать эксперименты по ее исследованию. Заметим, что большинство опций выбиралось нами по умолчанию.

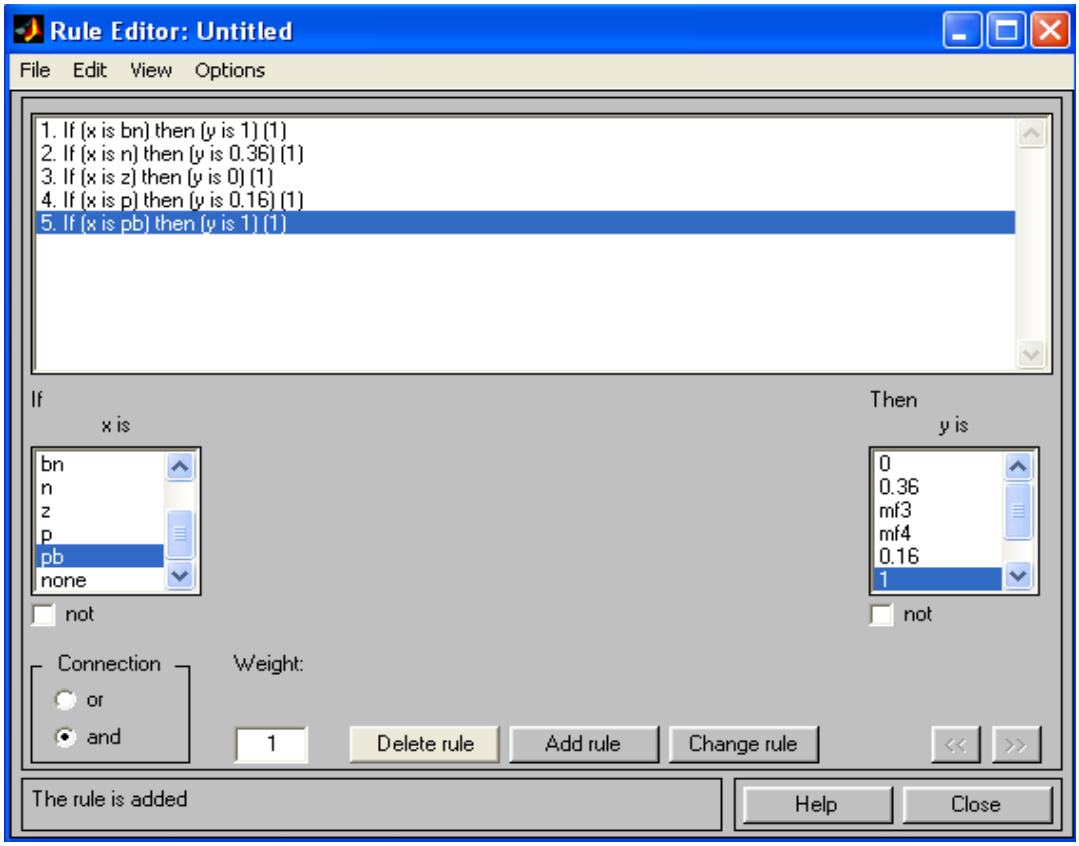


Рис. 2.5. Окно редактора правил

10. Предварительно сохраним на диске (используя пункты меню *File/Save to disk as...*) созданную систему под каким-либо именем, например, *Proba*.

11. Выберем позицию меню *View*. Как видно из выпадающего при этом подменю, с помощью пунктов *Edit membership functions* и *Edit rules* можно совершить переход к двум выше рассмотренным программам – редакторам функций принадлежности и правил (то же можно сделать и нажатием клавиш *Ctrl*+2 или *Ctrl*+3). Но сейчас нас будут интересовать два других пункта – *View rules* (Просмотр правил) и *View surface* (Просмотр поверхности). Выберем пункт *View rules*, при этом откроется окно (см. рис. 2.6) еще одной программы – просмотра правил (*Rule Viewer*).

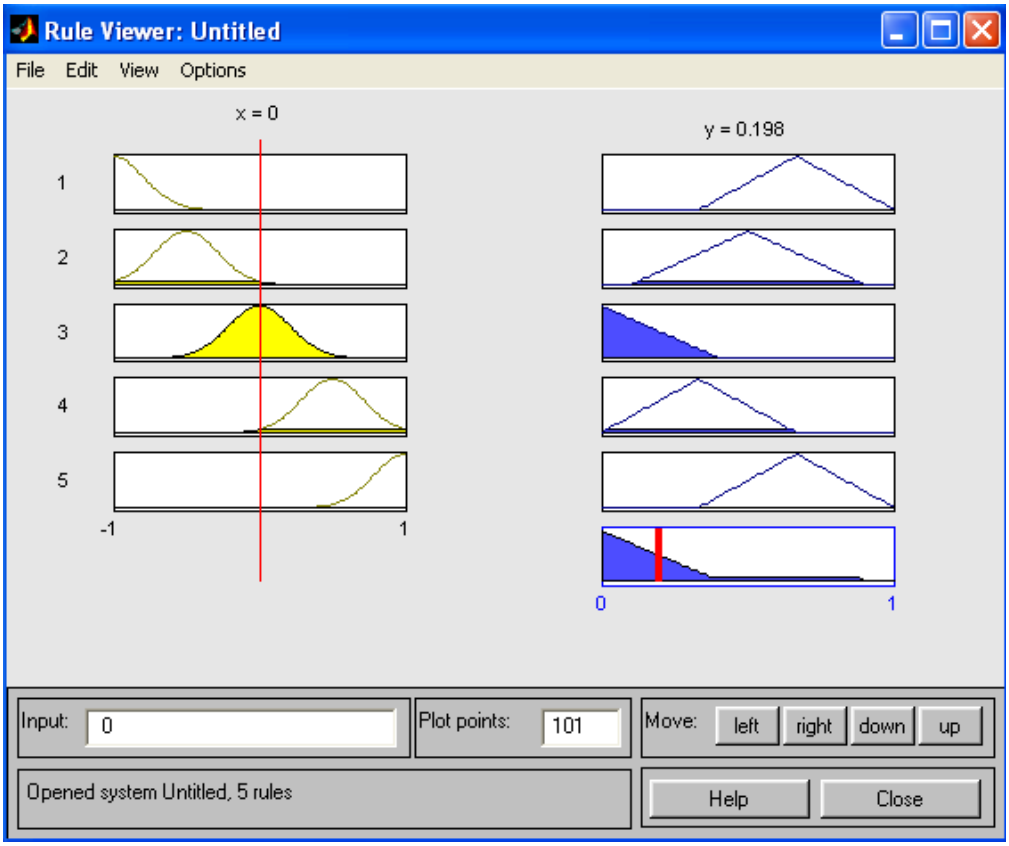


Рис. 2.6. Окно просмотра правил

12. В правой части окна в графической форме представлены функции принадлежности аргумента ж, в левой – переменной выхода *y* с пояснением механизма принятия решения. Красная вертикальная черта, пересекающая графики в правой части окна, которую можно перемещать с помощью курсора, позволяет изменять значения переменной входа (это же можно делать, задавая числовые значения в поле *Input* (Вход)), при этом соответственно изменяются значения *y* в правой верхней части окна. Зададим, например, *x* – 0.5 в поле *Input* и нажмем затем клавишу ввода (*Enter*). Значение *y* сразу изменится и станет равным 0.202. Таким образом, с помощью построенной модели и окна просмотра правил можно решать задачу интерполяции, т.е. задачу, решение которой и требовалось найти. Изменение аргумента путем перемещения красной вертикальной линии очень наглядно демонстрирует, как система определяет значения выхода.

Закроем окно просмотра правил и выбором пункта меню

View/View surface перейдем к окну просмотра поверхности отклика (выхода), в нашем случае – к просмотру кривой *y(x)* (см. рис. 2.7). Видно, что смоделированное системой по таблице данных (табл. 2.1) отображение не очень-то напоминает функцию x^2. Ну что ж, ничего удивительного в этом

нет: число экспериментальных точек невелико, да и параметры функций принадлежности (для *x)* выбраны, скорее всего, неоптимальным образом. Ниже мы рассмотрим возможность улучшения качества подобной модели.

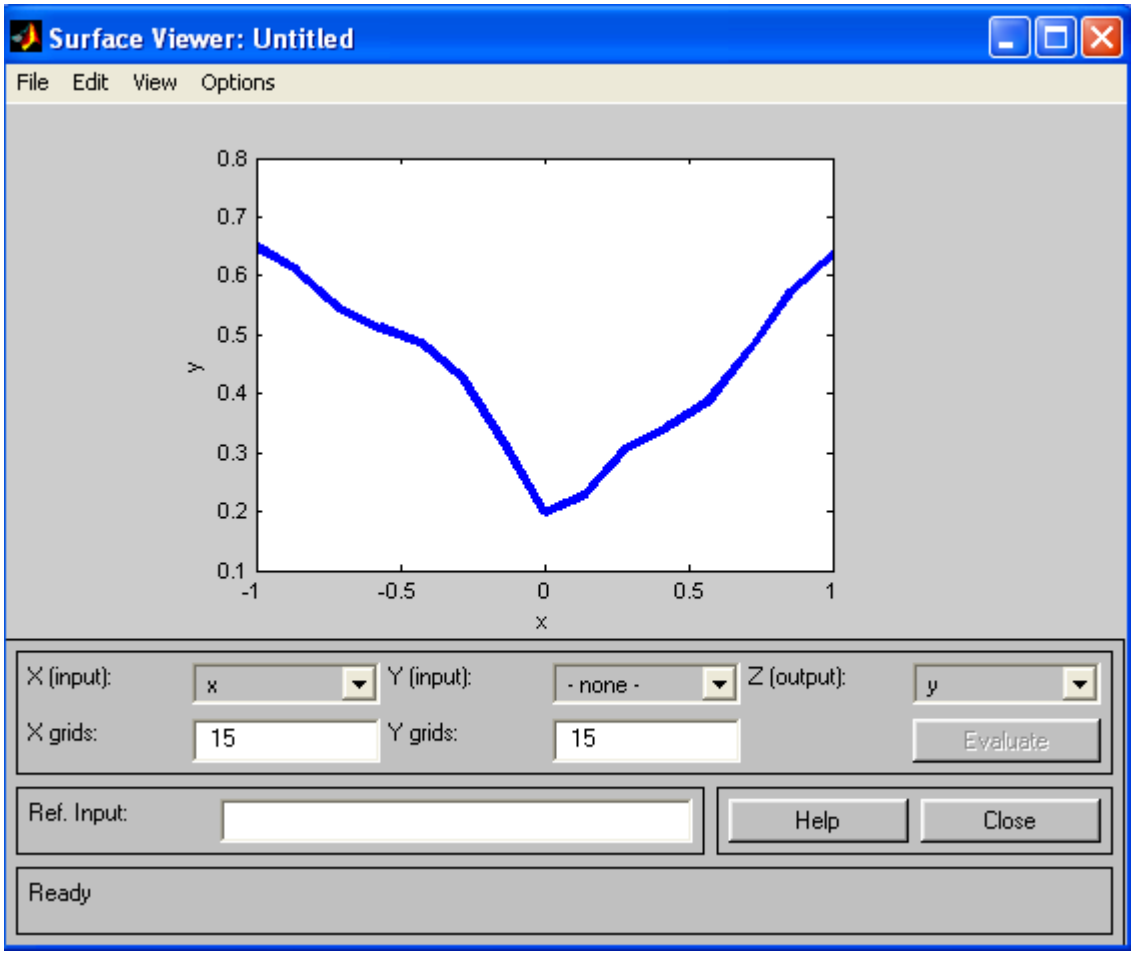


Рис. 2.7. Окно просмотра поверхности отклика

В заключение рассмотрения примера отметим, что с помощью вышеуказанных программ редакторов на любом этапе проектирования нечеткой модели в нее можно внести необходимые коррективы, вплоть до задания какой-либо особенной пользовательской функции принадлежности. Из опций, устанавливаемых в FIS-редакторе по умолчанию при использовании алгоритма Sugeno, можно отметить:

– логический вывод организуется с помощью операции умножения (*prod*);

– композиция – с помощью операции логической суммы (вероятностного ИЛИ, *probor*);

– приведение к четкости – дискретным вариантом центроидного метода (взвешенным средним, *wtaver*). Используя соответствующие поля в левой нижней части окна *FIS*-редактора, данные опции можно, при желании, изменить.

Индивидуальное задание

Сконструируйте нечеткую систему, отображающую зависимость между переменными *x* и *y*, заданную с помощью табл. 2. По результатам работы определить тип кривой.

Варианты заданий

Таблица 2.

	Значение аргумента и функции					
1	x	-1	-0.5	0	0.2	1
	y	1	0.25	0	0.4	1
2	x	-1	-0.6	0.2	0.4	1
	y	-1	-1.67	5	2.5	1

3	x	-1	-0.5	0	0.3	1
	y	-1	-0.13	0	0.27	1
4	x	-1	-0.6	0	0.3	1
	y	0	0.8	1	0.95	0
5	x	-1	-0.5	0	0.2	1
	y	1	-0.125	0	0.008	1
6	x	-1	-0.6	0.2	0.4	1
	y	0	-0.64	-0.96	-0.84	0
7	x	-1	-0.5	0	0.3	1
	y	-3	-2	-1	-0.4	1
8	x	-1	-0.6	0	0.3	1
	y	0.5	0.09	0	0.0225	0.5
9	x	-1	-0.5	0	0.2	1
	y	0.5	0.03125	0	0.0008	0.5
10	x	-1	-0.6	0.2	0.4	1
	y	-1	2.78	25	6.25	1
11	x	-1	-0.5	0	0.3	1
	y	-1	1.8	3	3.6	5
12	x	-1	-0.6	0	0.3	1
	y	-1	-0.216	0	0.027	1
13	x	-1	-0.5	0	0.2	1
	y	-2	-1.5	-1	-0.8	0
14	x	-1	-0.6	0.2	0.4	1
	y	-2	-1.2	0.4	2.5	1
15	x	-1	-0.5	0	0.3	1
	y	0	0.25	0.5	0.65	1
16	x	-1	-0.5	0	0.3	1
	y	-1	-1.75	-1	-0.31	0

Лабораторная работа №8. Часть 1. Моделирование нечеткой системы управления вентилятором

Цель работы:

– изучение основных определений теории нечетких множеств и теории нечеткого логического вывода;

– ознакомление с составом и возможностями инструментария нечеткой логики *Fuzzy Logic Toolbox*, входящего в пакет программ *MATLAB*;

– приобретение практических навыков работы в пакете *Fuzzy Logic Toolbox* и создание нечеткой модели управления.

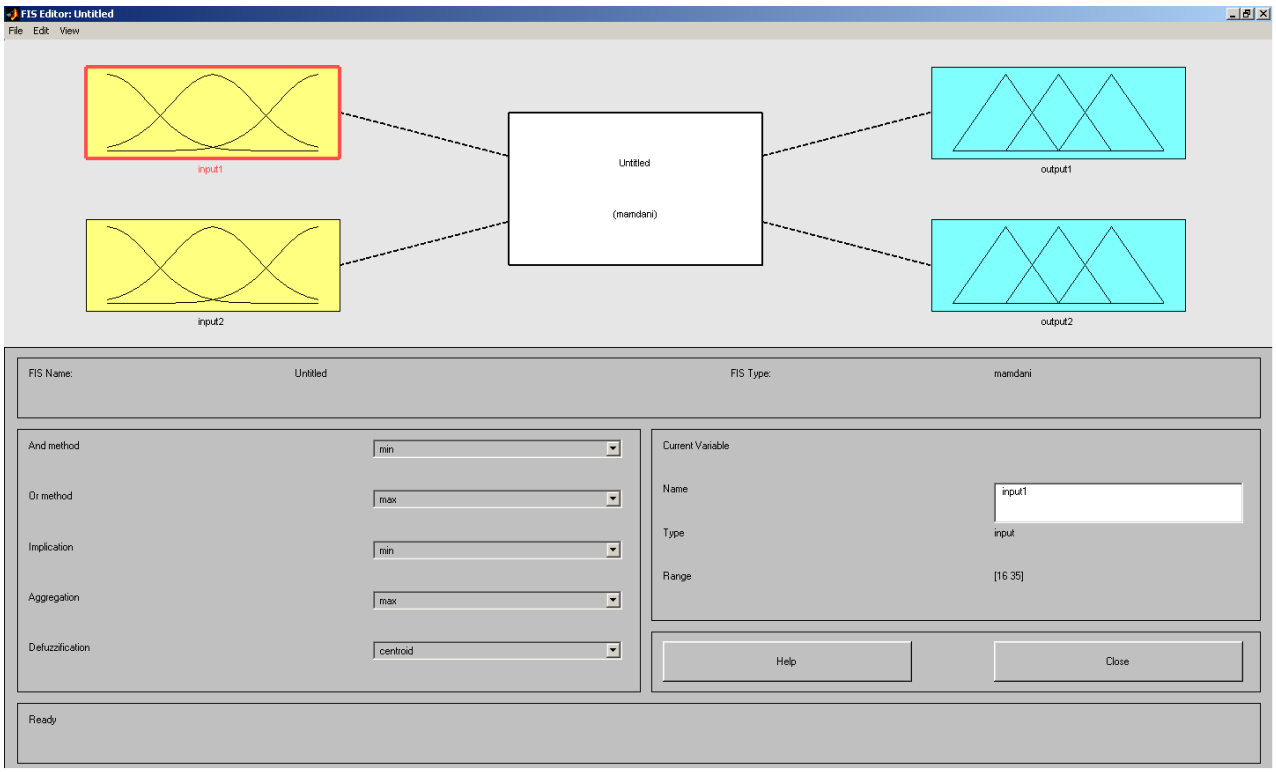
Ход работы

1. Постановка задачи

Определиться с количеством входных, выходных параметров, получить структурную схему системы управления. Реализовать модель управления.

2. Построение модели

Создадим двумерную модель кондиционера:



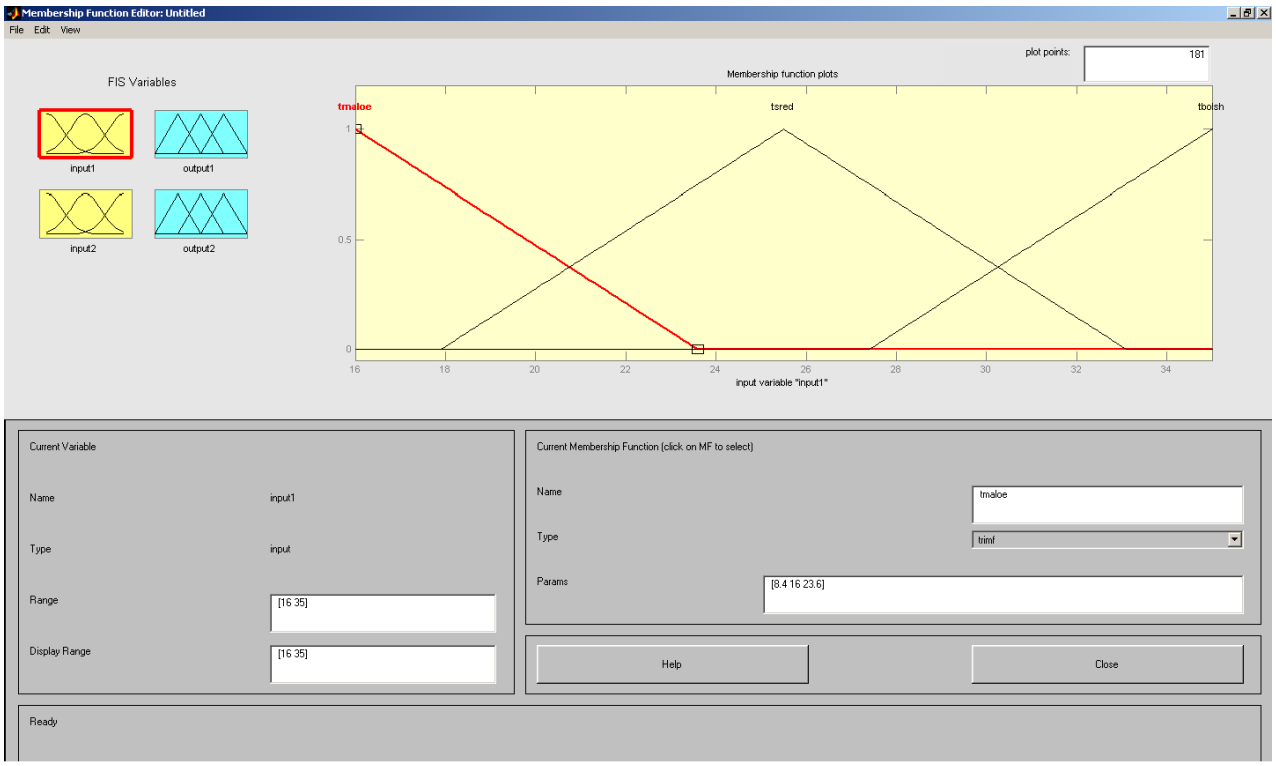
Входные параметры: температура внутри помещения и скорость изменения температуры в помещение

Выходные параметры: скорость вращения вентилятора и расход охлаждающей жидкости.

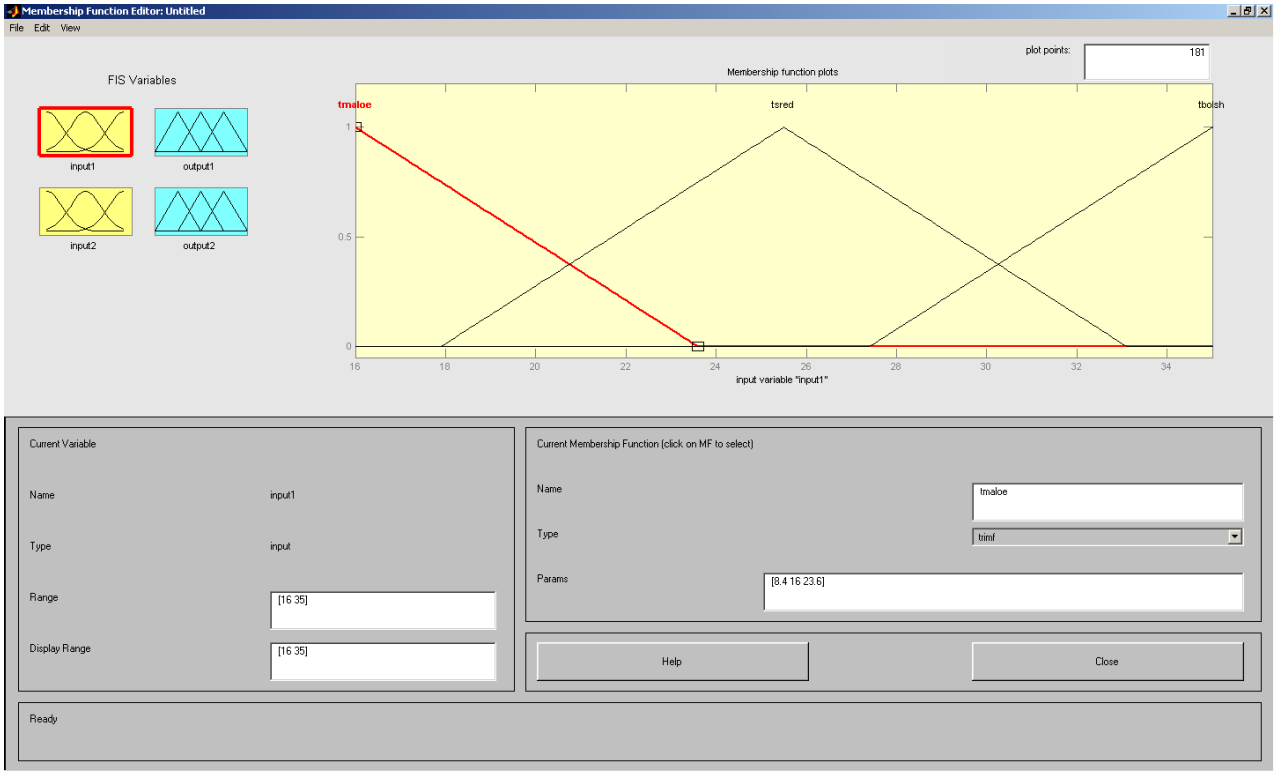
3. Создание термов

Для каждого параметра создадим термы:

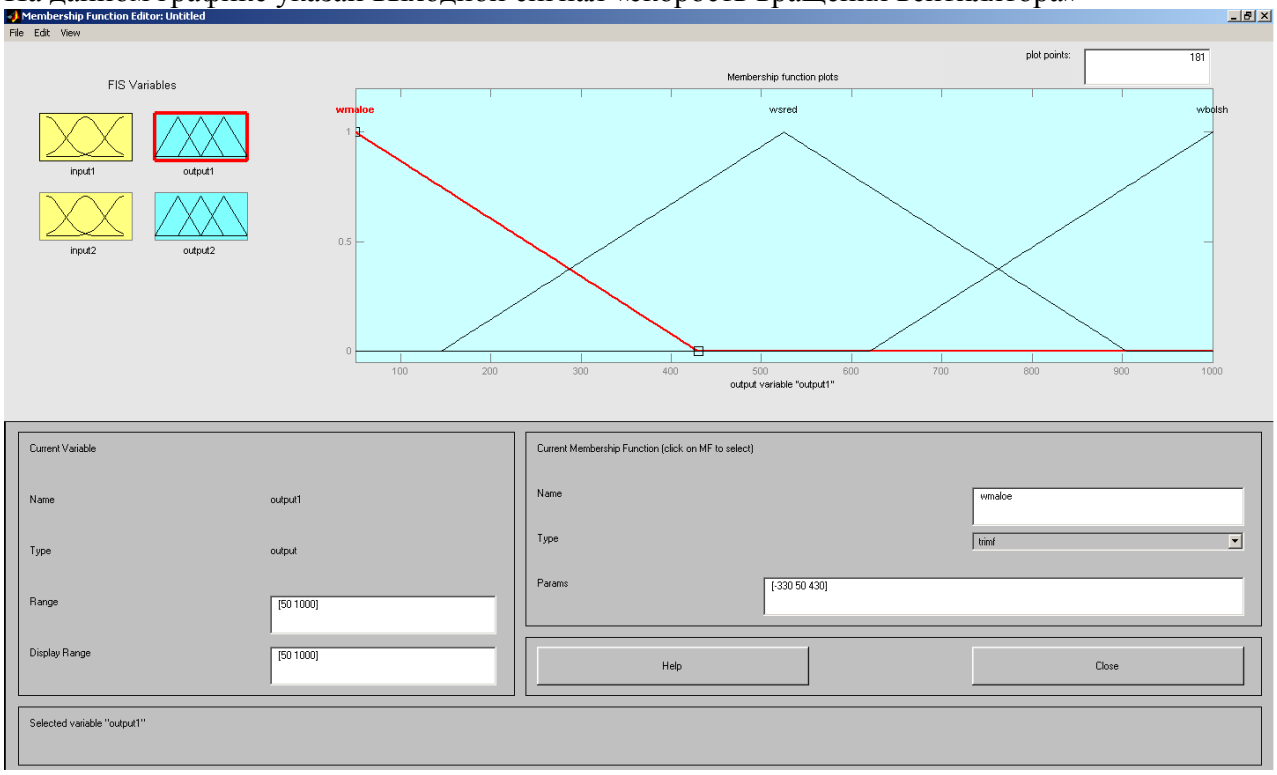
На данном графике указан входной сигнал «температура внутри помещения»



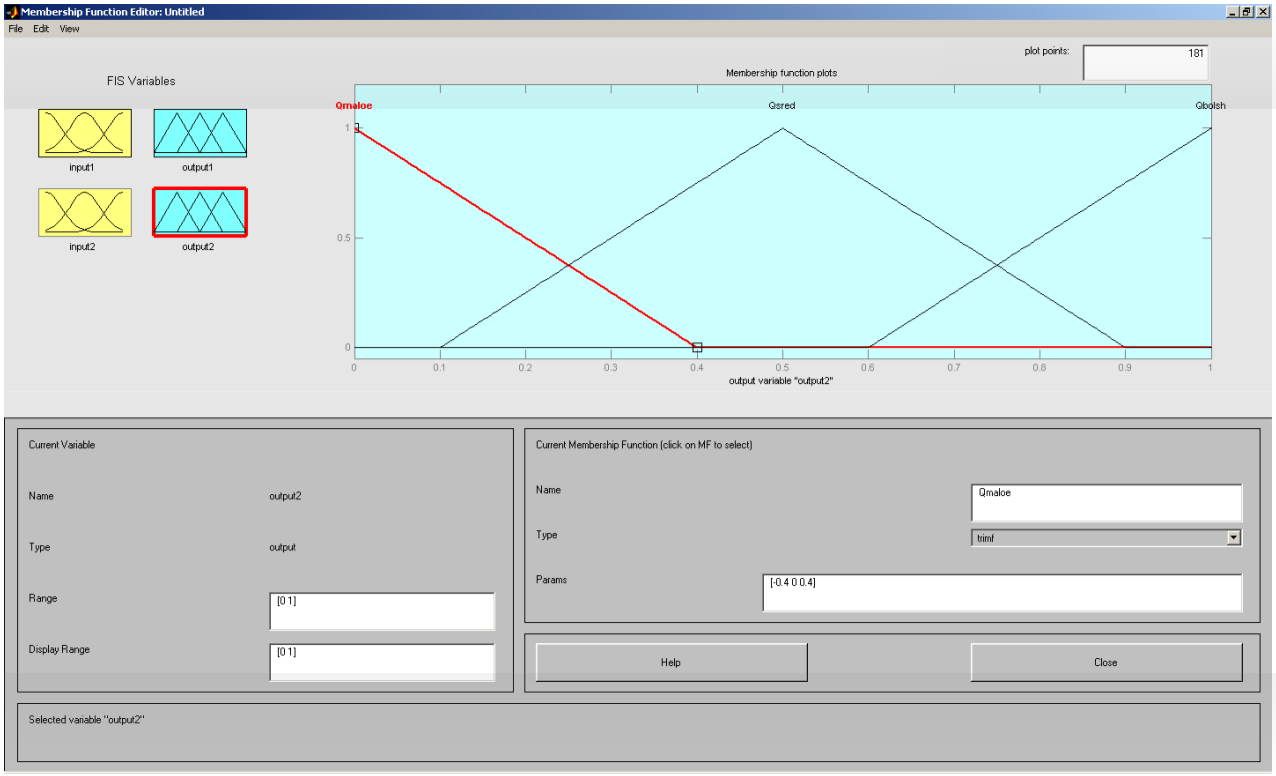
На данном графике указан входной сигнал «скорость изменения температуры в помещение»



На данном графике указан выходной сигнал «скорость вращения вентилятора»

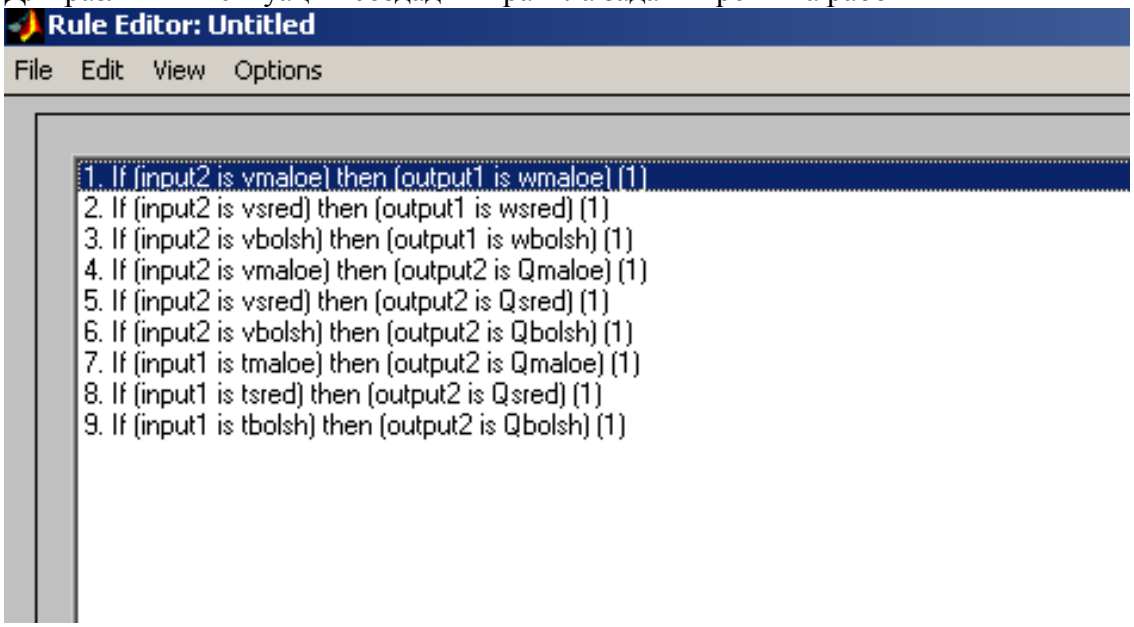


На данном графике указан выходной сигнал «расход охлаждающей жидкости»

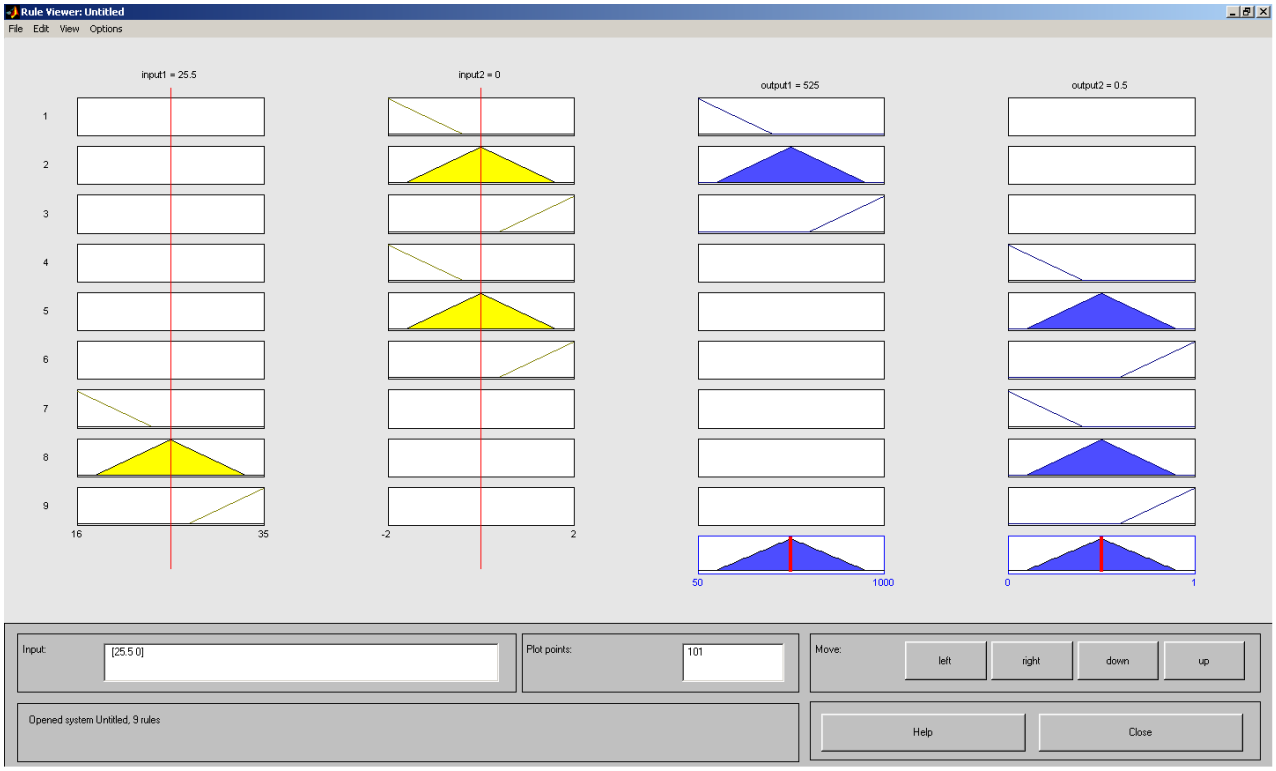


4. *Создание правил*

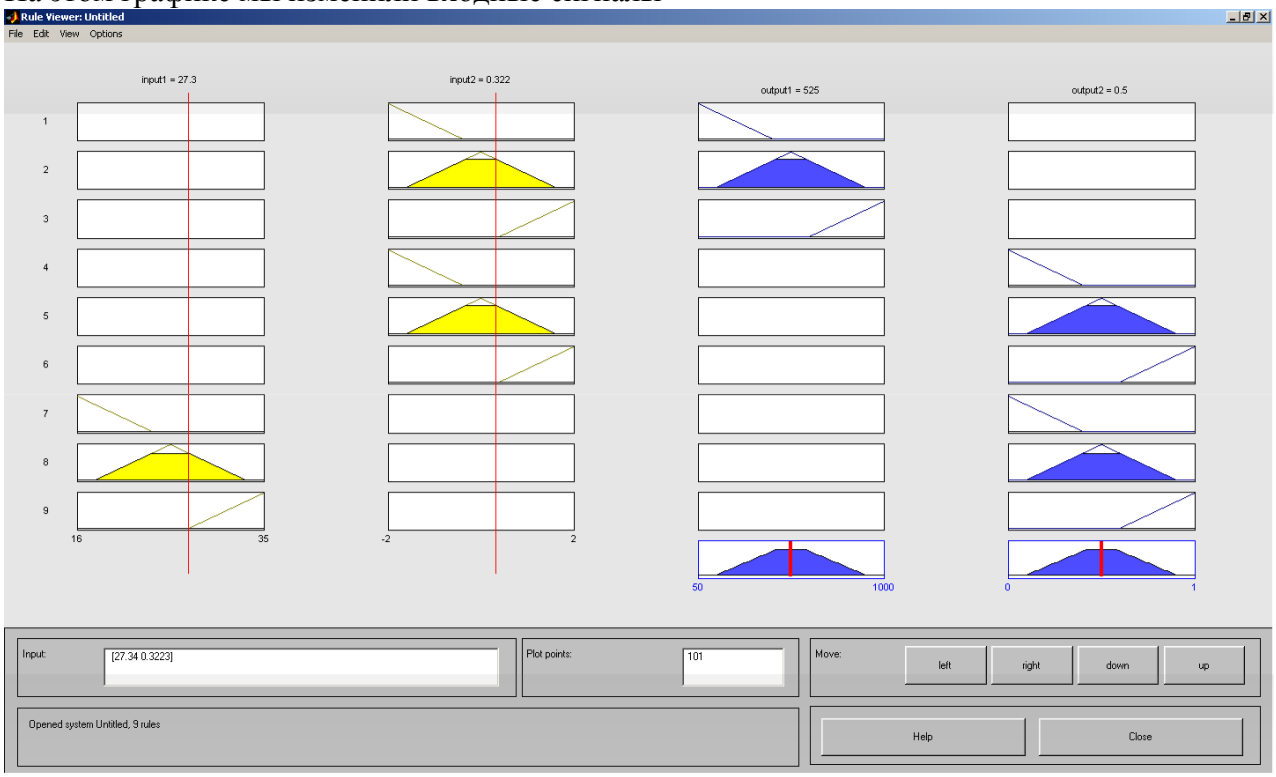
Для различных ситуаций создадим правила задания режима работы



Эти правила можно представить графически:

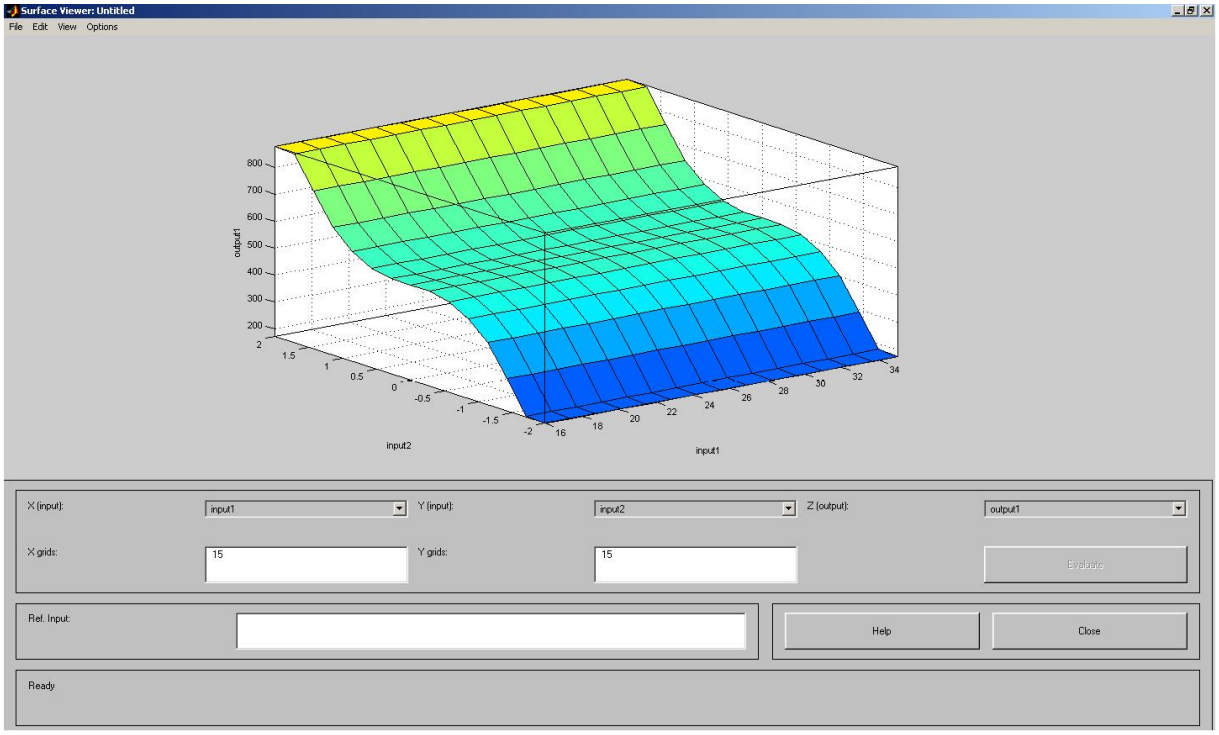


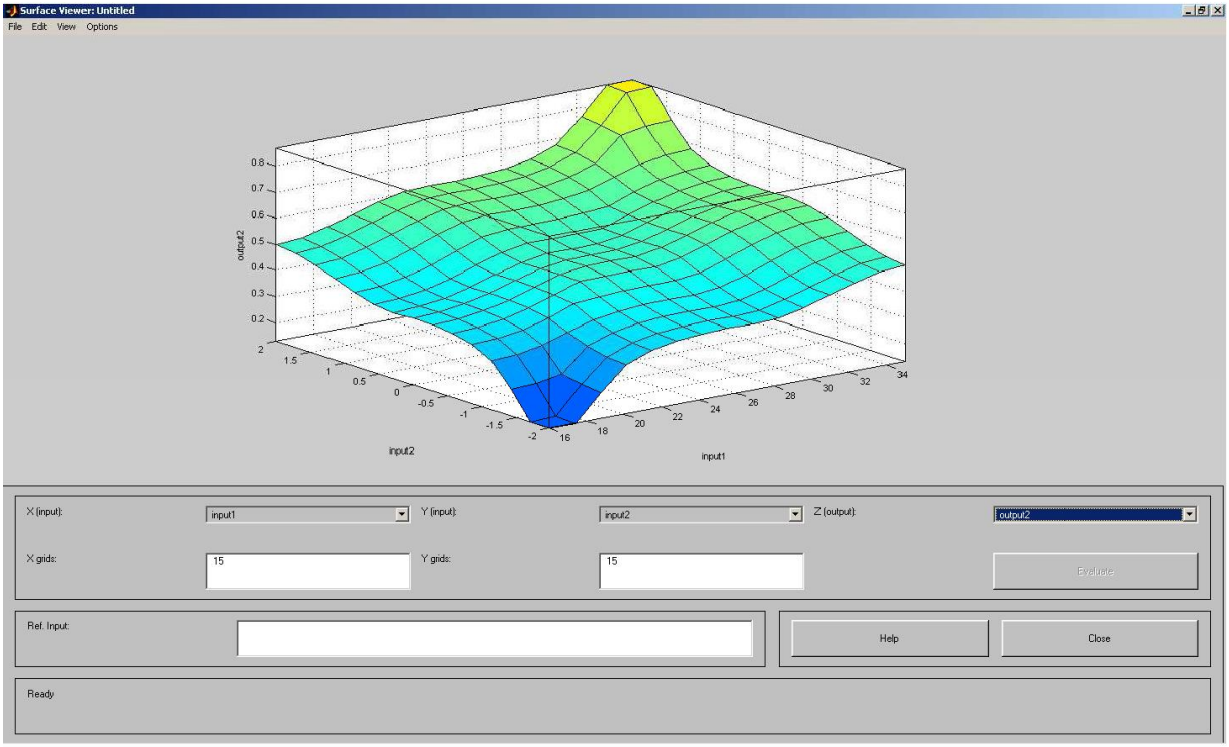
На этом графике мы изменили входные сигналы



5. Анализ полученной модели

Построим графики зависимости скорости вращения вентилятора, температуры в помещение и скорости изменения температуры.





Содержание отчета

1. Цель работы.

2. Описание нечеткой модели управления (шаги реализации, результирующие графики)
3. Провести исследования результатов моделирования при различных входных условиях (Привести таблицы значений, поверхности отклика).
4. Выводы по работе.

Лабораторная работа № 8. Часть 2. Нечеткая экспертная система с алгоритмом вывода Mamdani

Цели и задачи работы: Построение нечеткой экспертной системы с алгоритмом вывода Mamdani.

Задание

1. Создать нечеткую экспертную систему с алгоритмом вывода Mamdani, которая должна оценить уровень работы предприятия общественного питания. Использовать 2 входа, 1 выход, 3 правила типа «если... то», «если... или…то».
2. Построить усложненный вариант нечеткой экспертной системы для оценки работы предприятия или другого объекта (число входов и правил должно соответствовать заданному варианту). Использовать правила типа «если... то», «если... или…то», «если... и…то». Самостоятельно предложить входные переменные и правила вывода.

Варианты заданий:

Оценка квартиры для бюро обмена жилья: 5 входов, 7 правил. Уровень обслуживания в пассажирском поезде: 4 входа, 6 правил. Оценка успеваемости студентов: 3 входа, 5 правил.

Методические указания

При решении п.1 могут быть заданы следующие инструкции.

1. Если сервис плохой или еда несвежая, то оценка низкая.
2. Если обслуживание хорошее, то оценка средняя.
3. Если сервис отличный или еда вкусная, то оценка высокая.

При создании новой системы нечеткого вывода по умолчанию создается система с алгоритмом вывода Mamdani. Качество обслуживания и еды будем оценивать по 5-балльной системе.

В пункте меню EditVariable/Input добавить второй вход (появится второй блок с именем input2). Однократным щелчком левой кнопкой мыши по блоку input1 заменить его имя на «service», input2 – на «food», output1 – на «grade».

Задать функции принадлежности. Для переменной «service» в полях Range и DisplayRange установить диапазон изменения и отображения этой переменной – от 0 до 5 (подтверждая ввод нажатием клавиши Enter). Удалить заданные по умолчанию функции принадлежности с помощью мыши. Через пункт меню Edit/AddMFs задать функции принадлежности гауссова типа (gaussmf) в количестве 3. Заменить их имена на «bad», «good» и «excellent».

Щелчком мыши по «food» задать диапазон изменения от 0 до 5. После удаления заданных по умолчанию функций принадлежности задать две функции принадлежности трапецеидальной формы trapfm с параметрами [0 0 1 3] и

[2 4 5 5] и именами «notfresh» и «nice».

Для выходной переменной «grade» указать диапазон [0 5], задать три функции принадлежности треугольной формы с именами «bad», «good», «excellent».

Конструирование правил реализовано в пункте меню Edit/Rules. В первом и третьем правилах использовать «or» (единица в скобках после каждого правила указывает его «вес», который нужно оставить равным 1). При вводе второго правила, где отсутствует переменная «food», выбрать опцию none.

Из пункта меню View/Rules вызывать окно графического отображения функционирования системы, где можно задавать значения входных переменных в соответствующих полях, ответ отображается в правой части окна. Можно также перемещать отметку шкалы с помощью мыши.

Из пункта меню View/Surface вызвать отображение двумерной функции оценки работы предприятия общественного питания.

Сохранить созданную систему на диске: **File/Export/Todisk**.

При выполнении задания п.2 нужно усложнить созданную экспертную систему: ввести заданное число правил и входов. Экспертная система должна быть достаточно продуманной и иметь практическое значение.

Используемые операторы и команды: fuzzy – редактор нечеткой системы вывода. Программное обеспечение: MATLAB

Лабораторная работа № 9. Часть 1. Работа Fuzzy Logic с блоками Simulink

Цель работы: приобретение навыков Работы Fuzzy Logic с блоками Simulink.

Теоретическая часть

Система нечёткого вывода, созданные тем или иным образом с помощью пакета Fuzzy Logic Toolbox, допускают интеграцию с инструментами пакета Simulink, что позволяет выполнять моделирование систем в рамках последнего. Рассмотрим это на примере контроля уровня воды в баке [3].

Пример 1. Контроль уровня воды в баке.

На рис. 1 изображен объект управления в виде бака с водой, к которому подходят две трубы: через одну трубу, снабженную краном, вода втекает в бак, через другую – вытекает. Подачу воды в бак можно регулировать, больше или меньше открывая кран. Расход воды является неконтролируемым и зависит от диаметра выходной трубы (он фиксирован) и от текущего уровня воды в баке. Если понимать под выходной (регулируемой) переменной уровень воды, а под регулирующим элементом – кран, то можно отметить, что подобный объект регулирования, с точки зрения его математического описания, является динамическим и существенно нелинейным.

Определим цель управления здесь как установление уровня воды в баке на требуемом (изменяющемся) уровне и попробуем решить соответствующую задачу управления средствами нечеткой логики.

Очевидно, в регулятор, обеспечивающий достижение цели управления, должна поступать информация о несоответствии (разности) требуемого и фактического уровней воды, при этом данный регулятор должен вырабатывать управляющий сигнал на регулирующий элемент (кран).

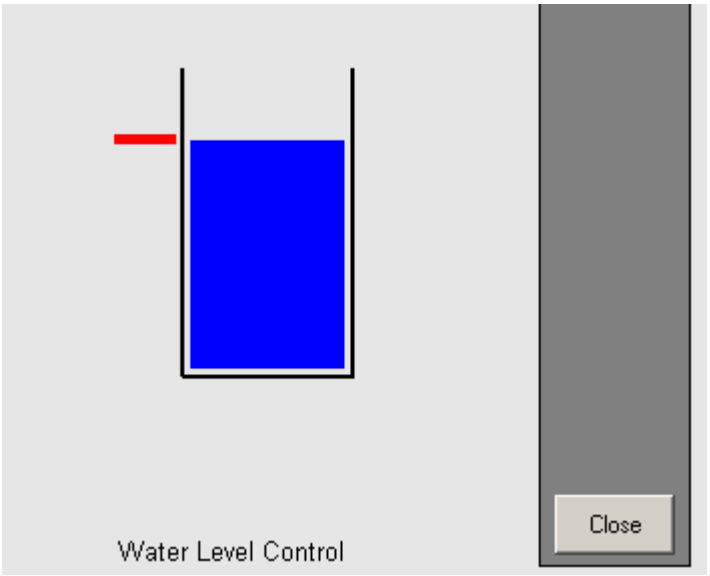


Рис. 1. Схематическое представление объекта управления (бака с водой).

В первом приближении функционирование регулятора можно описать набором из следующих правил:

1. If (level is okay) then (valve is no_change) (1)
2. If (level is low) then (valve is open_fast) (1)
3. If (level is high) then (valve is closefast) (1)
4. If (level is okay) and (rate is positive) then (valve is close_slow) (1)
5. If (level is okay and (rate is negative) then (valve is openslow) (1),

что в переводе означает:

1. Если (уровень соответствует заданному), то (кран без изменения) (1)
2. Если (уровень низкий), то (кран быстро открыть) (1)
3. Если (уровень высокий), то (кран быстро закрыть) (1)
4. Если (уровень соответствует заданному) и (его прирост – положительный), то (кран надо медленно закрывать) (1)
5. Если (уровень соответствует заданному) и (его прирост – отрицательный), то (кран надо медленно открывать) (1)

Модель системы управления уровнем воды в баке с нечетким регулятором, основанным на приведенных правилах, является одной из демонстрационных моделей пакетов Fuzzy Logic и Simulink.

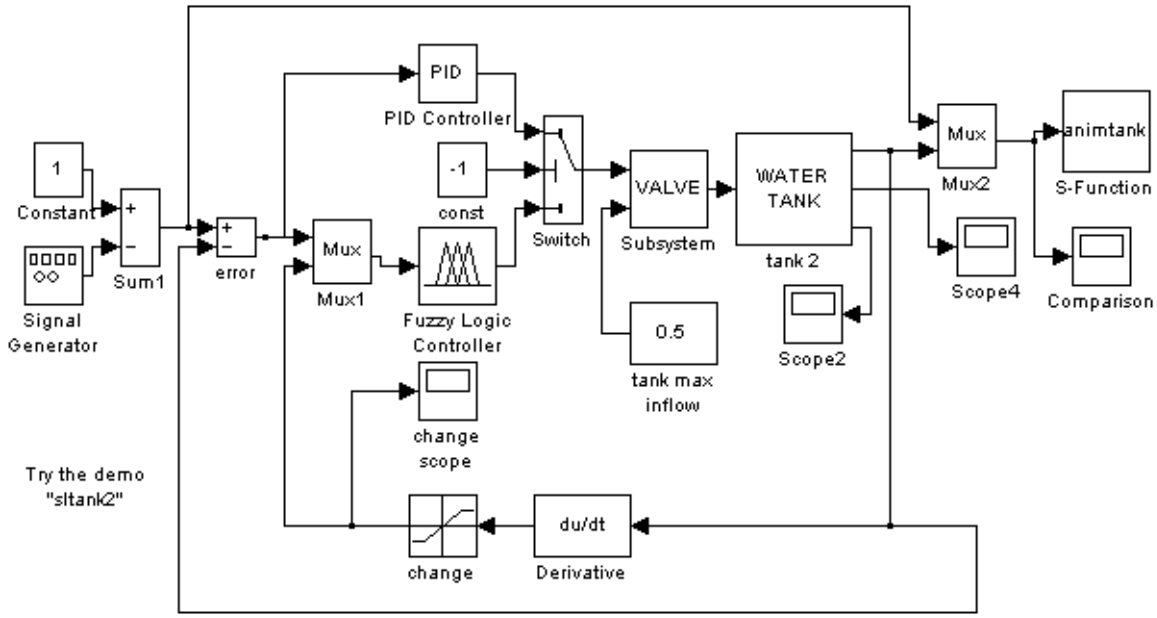


Рис. 2. Блок-диаграмма модели системы управления уровнем воды в баке нечётким регулятором

Ее можно вызвать из командной строки командой **sltank,** что приведет к открытию окна Simulink с блок-диаграммой указанной модели (рис. 2).

Одновременно блок Fuzzy Logic Controller будет сопоставлен с системой нечеткого вывода, записанной в файле tank.fis.

Для изучения процесса функционирования системы необходимо нажать кнопку **Start** панели инструментов блок-диаграммы модели и дважды щелкнуть левой кнопкой мыши по блоку Comparison (Сравнение). В появившемся окне этого блока будут показаны изменяющиеся во времени сигналы заданного (последовательность импульсов прямоугольной формы) и фактического уровней воды (рис. 3). Как видно, переходный процесс в системе имеет апериодическую форму и заканчивается достаточно быстро, т. е. качество регулирования следует признать хорошим.

Построение нечеткой модели с использованием блоков Simulink.

Для построения какой-то собственной моделирующей системы с использованием средств нечеткой логики и блоков Simulink рекомендуется просто скопировать блок Fuzzy Logic Controller из рассмотренной системы sltank (или какого-либо другого демонстрационного примера MATLAB) и поместить его в

блок-диаграмму разрабатываемой системы. Из командной строки командой **fuzblock** можно также открыть библиотеку нечетких блоков пакета Simulink, которые могут быть использованы точно так же, при необходимости – с дополнительным редактированием.

Отметим, что функционирование указанных блоков осуществляется с использованием системной S-функции **sffis.mex.** Запись этой функции такова:

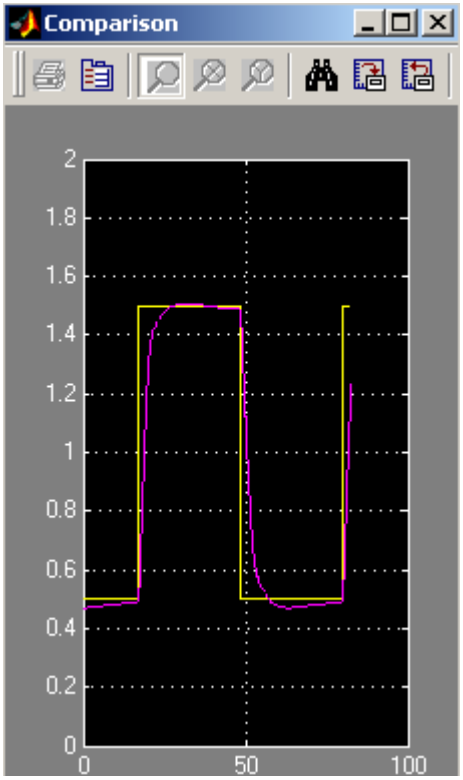


Рис. 3. Результаты моделирования системы управления с нечётким регулятором.

output = sffis(t,x,u,nag,nsmat),

где **output** – выход нечеткого регулятора, **t, x** и **flag** – стандартные аргументы системной функции Simulink, **fismat** – идентификатор (имя) нечеткой системы вывода, и входной сигнал (вектор входных сигналов) регулятора.

По смыслу данная функция аналогична рассмотренной выше функции **evalfis,** но она оптимизирована для работы в среде Simulink.

Демонстрационные примеры работы с пакетом Fuzzy Logic Toolbox.

Для ознакомления с пакетом Fuzzy Logic Toolbox можно использовать следующие функции (команды) в режиме командной строки:

defuzzdm – обзор методов приведения к четкости (дефазификации),

fcmdemo – демонстрация алгоритма кластеризации Fuzzy c-means (2-D графика),

fuzdemos – демонстрация графического интерфейса пакета Fuzzy Logic,

gasdemo – демонстрация использования аппарата гибридных сетей для решения задачи о выборе автомобиля, наиболее экономичного по расходу топлива,

juggler – демонстрация системы жонглирования мячом с использованием нечеткого регулятора,

invkine – демонстрация нечеткого управления движением роботаманипулятора,

irisfcm – демонстрация алгоритма кластеризации Fuzzy c-means,

noisedm – демонстрация решения задачи фильтрации на основе методов нечеткой логики,

slbb – демонстрация задачи «шар на качелях»,

slcp – демонстрация нечеткой системы управления перевернутым маятником,

sltank – демонстрация системы управления уровнем воды в баке с нечетким регулятором,

sltankrule – то же, что в предыдущем случае, но с дополнительным просмотром нечетких правил,

sltbu – демонстрация функционирования нечеткой системы управления грузовиком.

С каждым из этих примеров связан M-файл, fis-файл или/и блок-диаграмма Simulink, запуск которых производится с помощью одной из указанных команд. Текст пояснений в примерах

– на английском языке. Данные примеры доступны и через главное меню MATLAB (пункт Help/Examples and Demos, раздел Toolboxes/Fuzzy Logic). Дополнительную информацию можно получить, используя команду **help fuzzy.**

Программа работы и методические указания

Опишите набором из правил функционирование регулятора, представленного в соответствующем демонстрационном примере (см. табл. 1), пред-

ставьте результат моделирования.

Таблица 1

Вариант	Демонстрационный пример
1	juggler
2	invkine
3	slbb
4	slcp
5	sltbu

Содержание отчета

- цель работы;
- задание;
- краткое описание действий по пунктам;
- графики по всем пунктам программы;
- выводы по работе.

Лабораторная работа № 9. Часть 2. Разработка гибридных интеллектуальных систем в среде Matlab

Цель работы: Исследование процесса разработки адаптивной системы нейро нечеткого вывода для аппроксимации зависимости, описываемой некоторой математической функцией.

Порядок выполнения работы

С помощью адаптивной сети нечеткого вывода аппроксимировать функцию: $y = 2x^2$.

Функция задается в виде пар чисел, которые записываются в файл с расширением *.dat* (рис. 1.)

Для создания гибридной модели, загружаем приложение *Fuzzy Logic Toolbox* пакета *MATLAB* и создаем модель типа *Sugeno,* имеющий один вход и один выход, как показано на рис. 2.

Во вкладке *Edit* выбираем редактор *Anfis,* нажимаем *Load Data* и указываем путь к созданному файлу данных. Результат перечисленных выше действий представлены на рис.2.

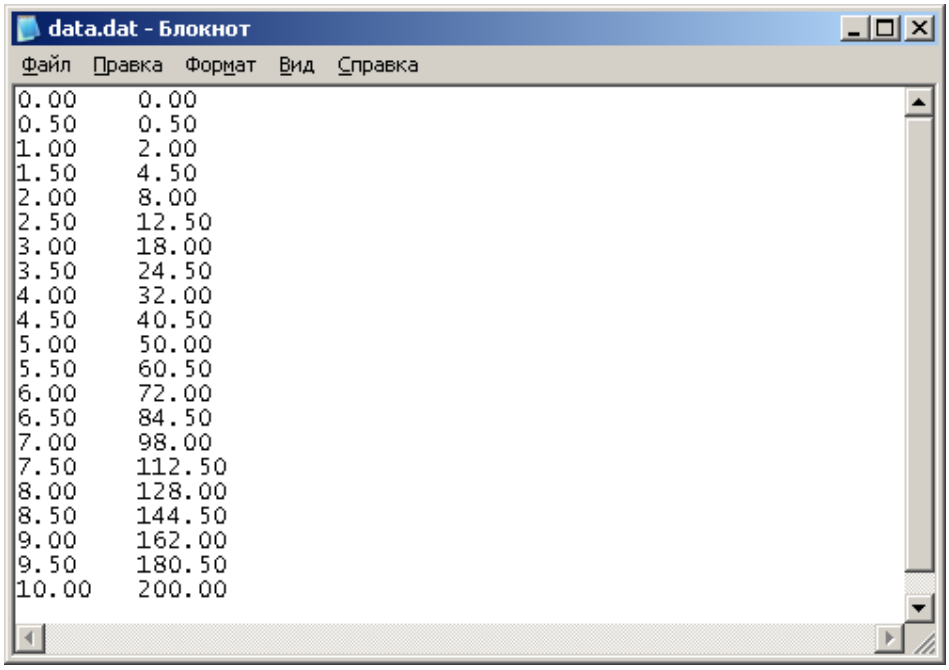


Рис..1. Исходные данные для обучения нейросети

Далее выбираем *Grid partition***,** появляется окно ввода параметров метода решетки (рис. 3), в котором нужно указать количество термов для каждой входной переменной и тип функций принадлежности для входных и выходной переменных. Выберем все, как указано на рис.4.3.

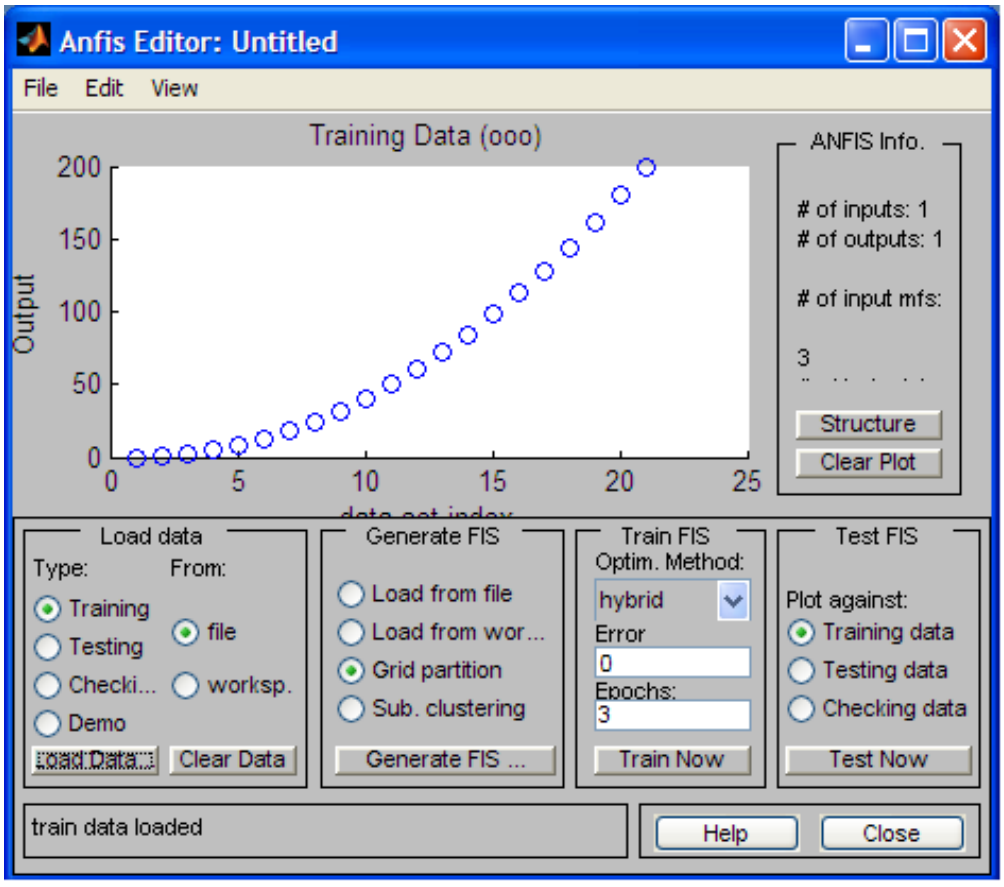


Рис. 2. Отображение данных

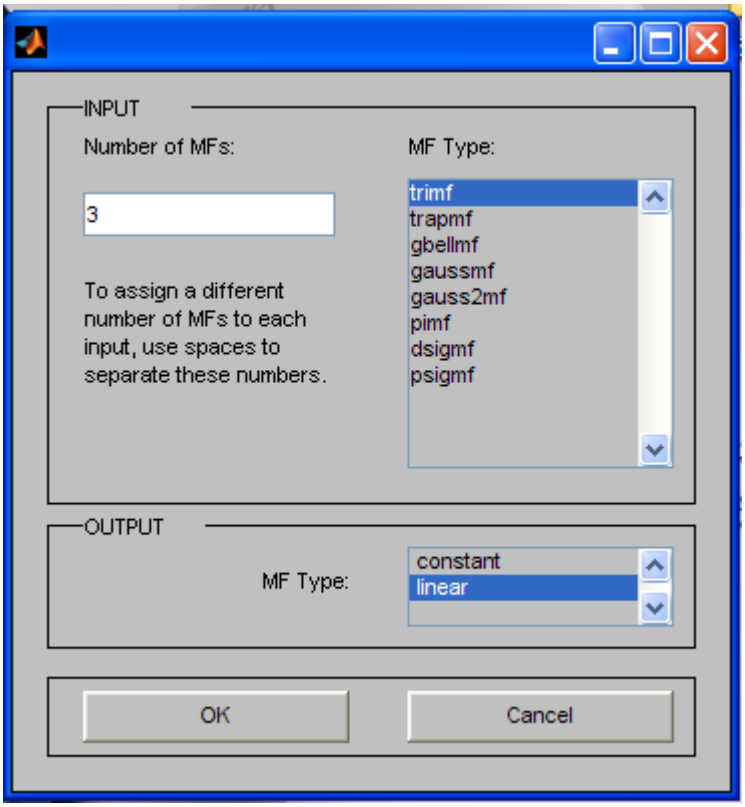


Рис.3. Окно ввода параметров метода решетки Структура нейросети имеет вид, представленный на рис.4.

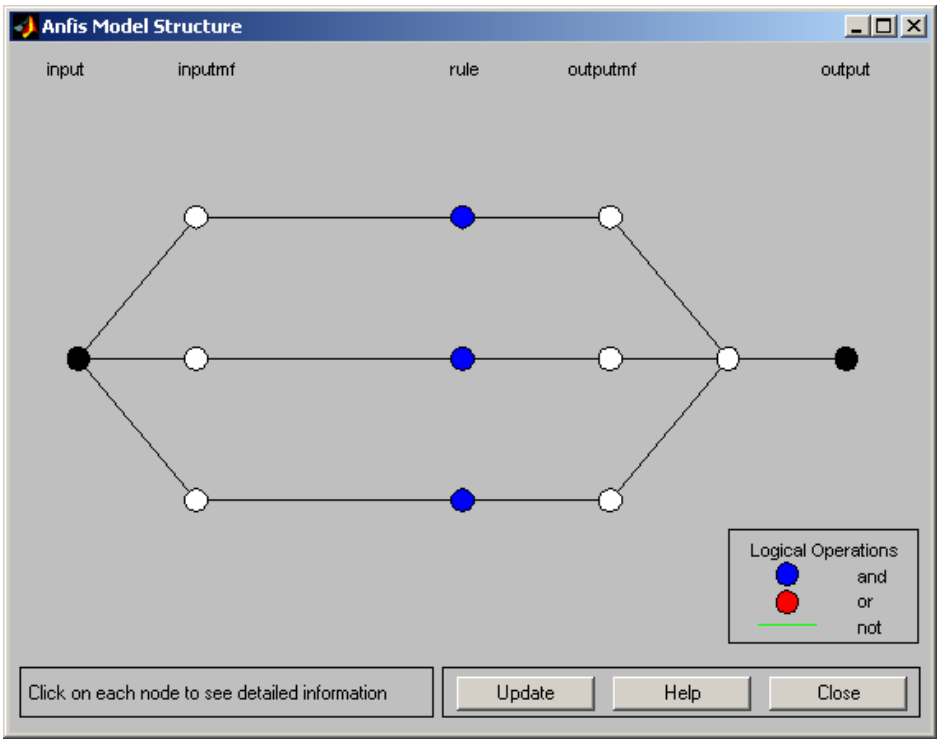


Рис.4. Структура нейро нечеткой сети

После этого, тренируем сеть, для этого в поле *Train FIS* вводим ошибку, количество эпох (рис. 5) и нажимаем кнопку *Train Now.*

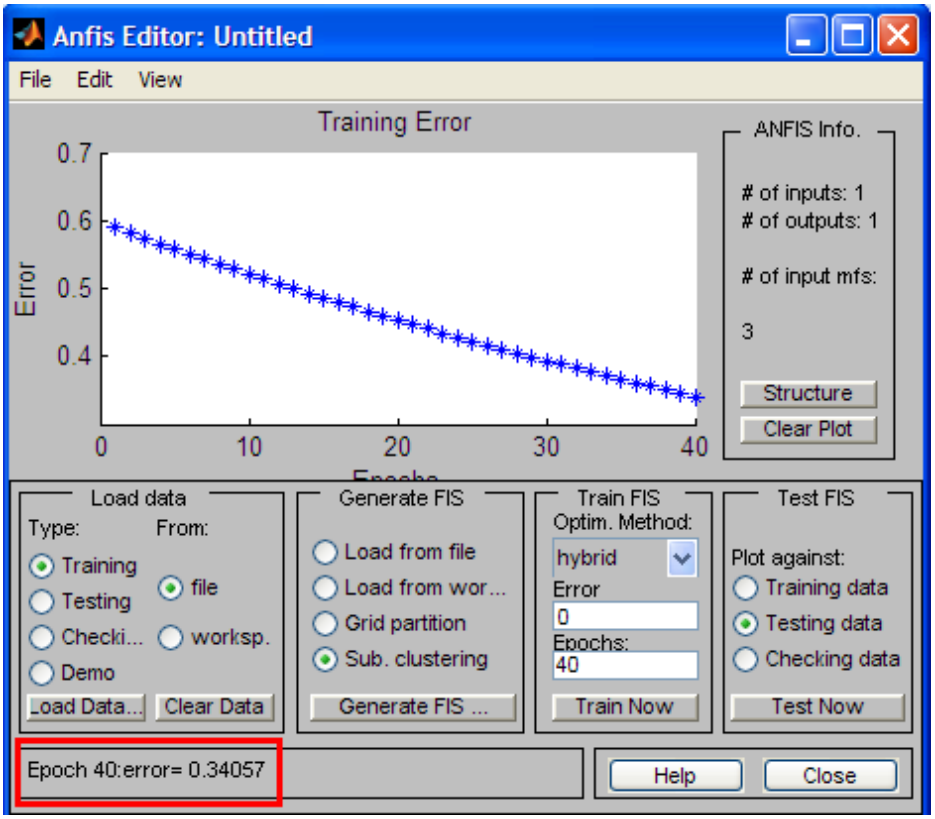


Рис. 5. Процесс обучения нейронной сети

После нажатия кнопки *Train Now*, в области визуализации отображается процесс обучения: число эпох – 40, значение ошибки – 0.34057.

Поверхность соответствующего нечеткого вывода и правила, сгенерированные в процессе обучения гибридной системы представлены на рис. 6, 7.

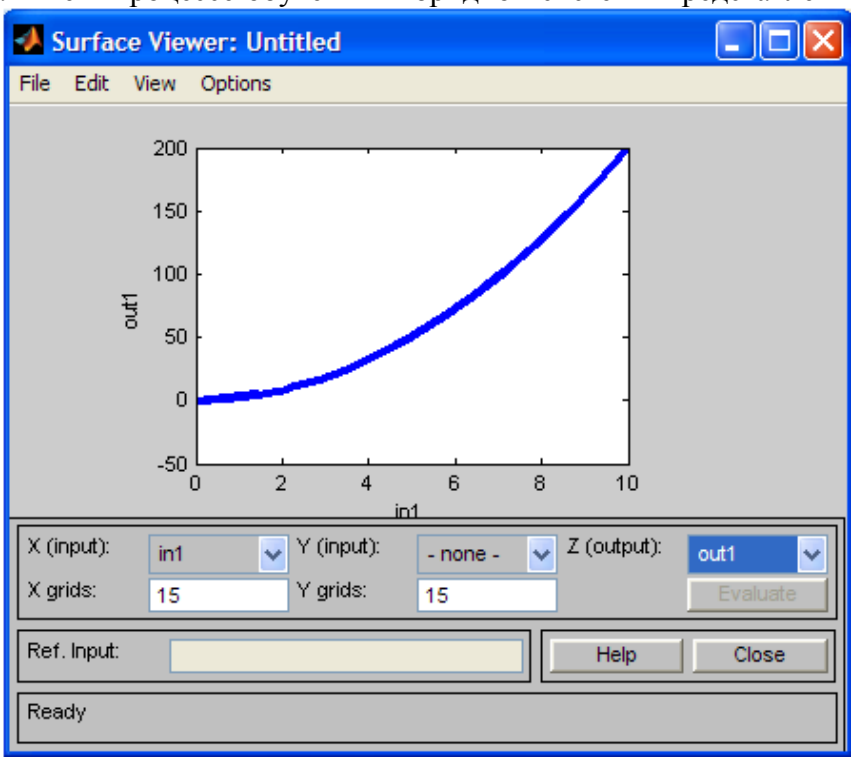


Рис.6. Поверхность системы нечеткого вывода

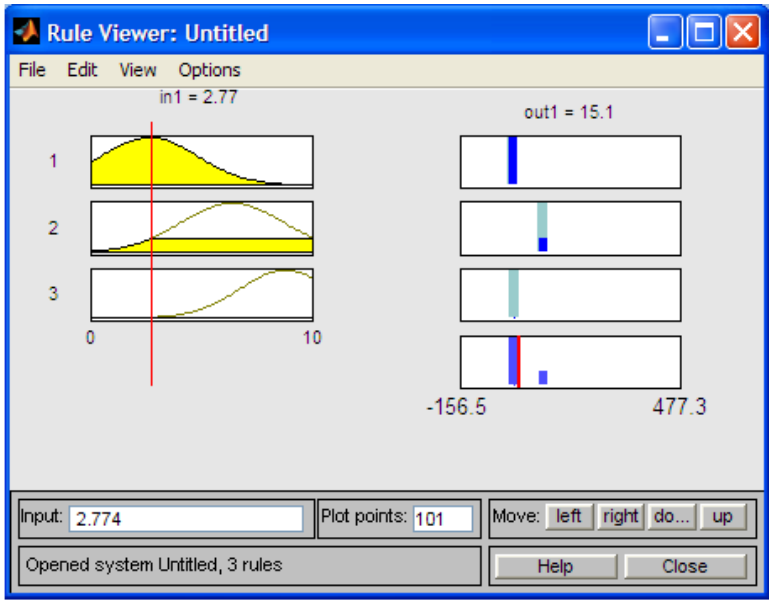


Рис.7. Нечеткие правила, полученные при обучении

Индивидуальное задание

Согласно варианту (табл. 4.1), создать гибридную модель аппроксимации функции. Провести обучение и сравнительный анализ результатов обучения. В случае необходимости (если отклонение от заданной функции составляет более 5%) провести коррекцию обучения, заменив количество и типа функций принадлежности, а также количества эпох обучения сети. Полученные результаты оформить в форме таблицы графиков ошибок.

Варианты заданий

Таблица 4.1.

$X(i)$	$Y(i)$ вариант									
	1	2	3	4	5	6	7	8	9	10
0	3,323	0,62	1,249	3,466	5,926	3,599	1,796	4,333	1,821	1,024
0.1	3,332	0,403	1,834	4,352	5,587	4,228	2,036	4,775	1,301	1,67
0.2	3,714	0,981	2,108	4,29	6,68	4,521	1,533	5,251	1,834	0,97
0.3	4,073	0,918	2,37	4,456	7,33	4,911	1,259	5,829	1,522	0,988
0.4	4,057	1,248	1,542	5,588	7,877	5,497	1,766	6,655	1,571	1,873
0.5	4,754	1,18	1,584	5,882	9,055	5,571	1,933	7,749	1,614	1,826
0.6	5,389	0,606	1,808	6,129	9,821	6,255	1,692	7,761	2,212	1,6
0.7	5,734	1,268	2,359	5,667	10,282	6,378	2,353	8,544	2,274	2,241
0.8	5,279	1,128	1,888	6,832	11,054	7,37	2,43	10,423	2,079	1,972
0.9	6,516	0,777	2,379	7,249	12,314	7,753	1,813	11,12	3,036	1,855
1	6,652	1,159	2,9	7,148	13,465	9,055	2,299	12,024	2,565	1,872
1.1	6,815	1,285	2,954	7,548	14,8	9,633	1,923	13,829	3,441	2,006
1.2	6,966	1,332	2,093	7,253	16,087	11,102	2,059	14,581	3,611	2,035
1.3	7,615	1,077	2,284	7,972	17,636	12,305	2,010	16,651	3,65	1,878
1.4	8,321	1,359	2,597	8,344	20,35	12,845	2,873	18,157	3,942	2,611
1.5	8,969	1,653	2,394	8,218	22,002	14,082	2,774	20,967	4,422	1,954
1.6	9,734	1,395	2,988	8,248	24,146	15,769	2,75	22,352	5	2,642
1.7	10,562	0,972	2,394	9,276	26,169	17,774	3,334	25,132	5,063	2,38
1.8	10,929	1,328	2,815	9,228	29,249	20,072	3,044	27,685	5,016	2,696
1.9	11,704	1,977	3,202	9,436	31,622	21,353	2,966	31,204	5,369	2,288

Продолжение табл.4.1.

$X(i)$	$Y(i)$ вариант									
	11	12	13	14	15	16	17	18	19	20
0	2,283	2,046	2,343	2,799	4,512	1,414	4,235	1,149	1,821	1,024
0.1	2,949	2,072	2,677	2,468	4,481	1,134	5,024	1,389	1,301	1,67
0.2	2,761	2,041	2,997	2,961	4,54	1,466	5,338	1,622	1,834	0,97

0.3	3,471	2,411	2,754	3,217	5,309	0,879	5,856	1,805	1,522	0,988
0.4	3,768	2,734	3,168	3,057	5,652	1,885	6,516	2,786	1,571	1,873
0.5	4,667	3,422	2,9	3,402	5,765	1,378	7,392	2,988	1,614	1,826
0.6	4,908	4,164	2,924	3,784	6,416	1,704	8,338	2,978	2,212	1,6
0.7	5,896	3,673	3,729	3,626	5,989	1,463	9,008	2,535	2,274	2,241
0.8	5,984	3,94	2,97	3,315	6,473	2,103	10,189	3,086	2,079	1,972
0.9	7,467	4,565	3,973	4,118	6,682	1,728	10,883	3,192	3,036	1,855
1	7,977	4,594	3,423	4,359	7,527	1,443	12,436	3,053	2,565	1,872
1.1	9,184	4,541	3,899	3,871	7,439	1,779	13,47	3,785	3,441	2,006
1.2	10,035	4,252	3,744	4,201	7,912	1,991	15,144	3,507	3,611	2,035
1.3	11,0,29	4,269	3,991	4,799	9,113	2,172	16,796	3,275	3,65	1,878
1.4	12,435	4,886	4,181	5,127	9,039	2,056	18,275	3,596	3,942	2,611
1.5	14,496	4,568	4,494	5,081	10,078	2,452	20,166	3,746	4,422	1,954
1.6	16,015	4,086	4,231	5,314	9,872	2,076	22,719	3,945	5	2,642
1.7	18,504	4,004	4,281	5,426	10,533	1,913	24,82	3,238	5,063	2,38
1.8	20,568	3,888	4,662	6,201	11,097	2,019	28,174	3,160	5,016	2,696
1.9	21,457	3,34	4,592	7	12,557	2,783	30,512	3,002	5,369	2,288

Содержание отчета

1. Цель работы.
2. Описание гибридной модели аппроксимации функции и анализ результатов аппроксимации и коррекции модели (таблицы, графики).
3. Выводы по работе.

Список литературы

1. Салмина, Н.Ю. Функциональное программирование и интеллектуальные системы : учебное пособие / Н. Ю. Салмина. — Томск : Томский государственный университет систем управления и радиоэлектроники, 2016. — 100 с. — ISBN 2227-8397. — Текст : электронный // Электронно- библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/72216.html

2. Трофимов, В. Б. Интеллектуальные автоматизированные системы управления технологическими объектами / В. Б. Трофимов, С. М. Кулаков. — М. : Инфра-Инженерия, 2016. — 232 с. — ISBN 978-5-9729-0135-7. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/51726.html

3. Архипов, С.Н.; Основы теории управления техническими системами Электронный ресурс : учебное пособие / С.Н. Архипов. - Новосибирск : Сибирский государственный университет телекоммуникаций и информатики, 2016. - 166 с. - Книга находится в базовой версии ЭБС IPRbooks.

4. Интеллектуальные системы управления организационно-техническими системами : [науч. изд.] / А.Н. Антамошиш, О.В. Близнова, А.В. Бобов и др. ; под ред. А.А. Большакова. - М. : Горячая линия-Телеком, 2006. - 160 с. : ил. - Прил.: с. 138-145. - Библиогр.: с. 146-154. - ISBN 5-93517-289-5